Universidad Autónoma de Madrid
Escuela Politécnica Superior

Máster Oficial en Bioinformática y Biología Computacional

Trabajo Fin de Máster

digitalDLSorteR: paquete de R para la deconvolución de muestras *bulk RNA-seq* basado en Redes Neuronales.

Autor: Diego Mañanes Cayero
Tutor: Fátima Sánchez Cabo
Co-tutor: Carlos Torroja Fugairiño
Ponente: Luis del Peso Ovalle

Septiembre 2020

digitalDLSorteR: paquete de R para la deconvolución de muestras *bulk RNA-seq* basado en Redes Neuronales.

Autor: Diego Mañanes Cayero
Tutor: Fátima Sánchez Cabo
Co-tutor: Carlos Torroja Fugairiño
Ponente: Luis del Peso Ovalle

Centro Nacional de Investigaciones Cardiovasculares (CNIC)
Unidad de Bioinformática
Septiembre 2020

Resumen

Resumen

El cáncer es una enfermedad multifactorial en la que intervienen una gran cantidad de variables. Entre ellas, la elevada diversidad de células inmunes que presentan los tumores sólidos ha sido centro de estudio durante las últimas dos décadas, ya que cumple un papel clave en la progresión de la enfermedad y la eficacia de los tratamientos. Para abordar su estudio, las tecnologías de secuenciación transcriptómica a nivel de célula única (*single-cell RNA-seq*, *scRNA-seq*) son fundamentales, ya que permiten conocer el estatus funcional de un gran repertorio de células y, por tanto, su identificación. Sin embargo, aún no es una metodología práctica para su aplicación sobre grandes cohortes de muestras debido a que presenta costes relativamente altos y protocolos de complejidad elevada. En su lugar, las tecnologías de secuenciación transcriptómica a nivel de tejido (*bulk RNA-seq*) siguen siendo más comunes a pesar de no tener en cuenta en qué medida contribuye cada tipo celular a los valores de expresión cuantificados. De ello, surge la necesidad de métodos computacionales capaces de estimar la proporción de tipos celulares presentes en los datos de expresión tisular, tarea conocida como deconvolución. Entre las herramientas publicadas, Torroja y Sanchez-Cabo (2019) presentaron digitalDLSorter, que se caracteriza por aprovechar la valiosa información que contienen los datos *scRNA-seq* para resolver el problema. Se basa en el uso de Redes Neuronales Profundas, algoritmos de Aprendizaje Automático cuyos resultados están revolucionando dicho campo durante los últimos años. Sin embargo, digitalDLSorter aún no contaba con una implementación que facilitase su uso por otros investigadores, ya que consistía en una *pipeline* compuesta por diferentes *scripts* que escribían en disco cada uno de los pasos realizados. Por ello, en el presente Trabajo Fin de Máster se ha desarrollado digitalDLSorteR, una nueva versión de la herramienta en forma de paquete de R. El objetivo es hacer más accesible su uso por otros investigadores tanto para la deconvolución directa de muestras mediante el uso de modelos pre-entrenados como para la construcción de nuevos modelos a partir de datos *scRNA-seq*. Además, con el fin de su puesta en práctica, se han analizado datos *scRNA-seq* procedentes de cáncer de mama para la caracterización de los tipos celulares presentes en ellos. Los modelos resultantes, capaces de estimar las proporciones de diferentes células inmunes en muestras procedentes de dicha enfermedad de forma precisa, se han integrado en el paquete en forma de modelos pre-entrenados, haciendo posible su uso desde la herramienta. Finalmente, han sido aplicados sobre datos *bulk RNA-seq* reales y se han analizado los resultados mediante correlaciones entre las proporciones predichas.

Palabras Clave

deconvolución redes neuronales profundas single-cell RNA-seq cáncer de mama

Abstract

Cancer is a multifactorial disease in which a large number of variables are involved. Among them, the high diversity of immune cells exhibited by solid tumors has been the focus of many studies over the last two decades, since it plays a key role in the progression of the disease and the effectiveness of treatments. In order to address its study, transcriptomic sequencing technologies at single cell level (single-cell RNA-seq, scRNA-seq) are fundamental, since they allow knowing the functional status of a great repertoire of cells and, therefore, their identification. However, it is not yet a practical method for its application on large cohorts of samples due to relatively high costs and high complexity protocols. Instead, bulk RNA-seq technologies are still more common despite the fact that they do not take into account the extent to which each cell type contributes to the quantified expression values. Hence, the need arises for computational methods capable of estimating the proportion of cell types present in tissue expression data, a task known as deconvolution. Among the published tools, the co-tutors of this project presented DigitalDLSorter, which is characterized by taking advantage of the valuable information contained in scRNA-seq data to solve the problem. It is based on the use of Deep Neural Networks, Machine Learning algorithms whose results are revolutionizing the field during the last years. However, digitalDLSorter did not yet have an implementation that would facilitate its use by other researchers, since it consisted of a pipeline composed by different scripts that wrote each of the steps performed on-disk. Therefore, in the present Master's Final Project digitalDLSorteR has been developed as a new version of the tool in the form of an R package. The aim is to make it more accessible for use by other researchers both for the direct deconvolution of samples through the use of pre-trained models and for the construction of new models from scRNA-seq data. In addition, in order to put it into practice, scRNA-seq data from breast cancer has been analyzed for the characterization of the cell types present in it. The resulting models, capable of accurately estimating the proportions of different immune cells in in breast cancer samples, have been integrated into the package as pre-trained models, making it possible to use them from the tool. Finally, they have been applied on bulk RNA-seq data and the results have been analyzed by correlations between the predicted proportions of the different cell types.

Key words

deconvolution deep neural networks single-cell RNA-seq breast cancer

Agradecimientos

Quiero darle las gracias a Fátima por darme la oportunidad de realizar mi Trabajo Fin de Máster en un entorno tan enriquecedor como lo es la Unidad de Bioinformática del CNIC. He aprendido muchas cosas nuevas durante los seminarios y las charlas de las comidas. También, y en especial, a Carlos, por ayudarme cuando lo he necesitado y enseñarme tantas cosas durante todo este tiempo, sin él es probable que no hubiera sido capaz de sacar este proyecto adelante.

En un plano más familiar, las personas a las que siento que debo agradecerles todo siempre que hago un trabajo de esta envergadura son mis padres. Sin ellos, yo no podría haber dedicado los últimos 5 años de mi vida a estudiar lo que me entusiasma.

Finalmente, pero no menos importante, quería dar las gracias a mi pareja Alba por el inmenso apoyo que me ha proporcionado durante estos meses de trabajo intenso. Ha sido un año muy duro y ella siempre ha estado ahí, espero poder devolver tanto algún día.

Índice general

Índice de Figuras

Índice de Tablas

Índice de Code Boxes

1 Introducción

1.1. Contexto y motivación del proyecto

1.1.1. Cáncer, micro-entorno tumoral y contexto inmune

En Biología, los tejidos son definidos como agrupaciones de células que presentan una serie de características similares y que, gracias a que están organizadas de una manera determinada, actúan juntas llevando a cabo funciones específicas (Alberts et al., 2016). Sin embargo, estas unidades no son idénticas entre sí, sino que presentan una gran diversidad por el hecho de que tanto el origen embrionario como el estado funcional que las caracterizan pueden ser diferentes. Esto obliga a estudiar los tejidos no como entidades homogéneas, sino como complejos ecosistemas constituidos por diferentes tipos celulares con múltiples relaciones. Este hecho cobra especial relevancia en el caso de las enfermedades complejas, es decir, aquellas cuyo origen y desarrollo no dependen de un único factor, sino de la combinación de diferentes variables como la genética o el entorno (Boyle et al., 2017). Conocer la composición celular de los tejidos afectados en estos casos es imprescindible, ya que esta variable ejerce un papel de vital importancia en el entorno en el que se desarrolla la enfermedad (Hansson y Libby, 2006; Yang, 2020).

El ejemplo más paradigmático de este tipo de situaciones es el cáncer. El tumor, como tejido biológico, debe ser analizado no como una masa homogénea compuesta únicamente por células malignas, sino como el resultado del sumatorio de diferentes tipos celulares cuyas complejas interacciones heterotópicas dan lugar a lo que se conoce como micro-entorno tumoral (Gerdes et al., 2014). De forma general, su composición consiste principalmente en las células tumorales *per se*, células estromales, endotelio, células del sistema inmune infiltradas (TILs (*Tumor Infiltrated Lymphocytes*), macrófagos, etc.) y una compleja matriz extracelular (Figura 1.1.a). Estos ingredientes, junto con sus interacciones, dan lugar a una serie de capacidades que definen la enfermedad conocidas como el sello de identidad del cáncer o *Hallmarks of Cancer* (Hanahan y Weinberg, 2000, 2011) (Figura 1.1.b). Hasta ahora, han sido descritos 10 sucesos comunes en mayor o menor medida entre los diferentes tipos de tumores: autosuficiencia de factores de crecimiento, invasión tisular y metástasis, proliferación potencial ilimitada, angiogénesis permanente, evasión de la apoptosis, desregulación del metabolismo, inestabilidad genómica, inflamación producida por el tumor y evasión del sistema inmune. Son estas características las que hacen del cáncer una enfermedad difícil de estudiar, ya que presenta un carácter multifactorial de gran complejidad.

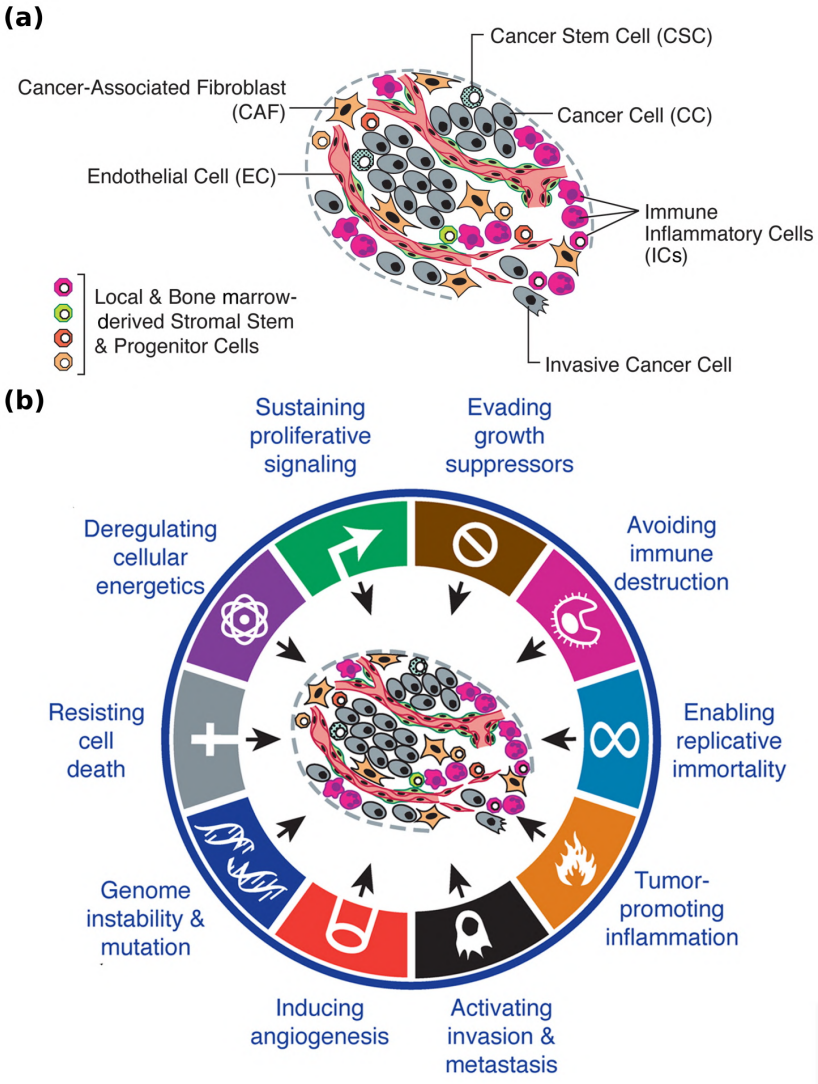

Figura 1.1: (a): Representación esquemática del micro-entorno tumoral. **(b):** Marcas de identidad del cáncer. Figuras tomadas de Hanahan y Weinberg (2011).

Teniendo en cuenta esta visión de la enfermedad, es innegable la transcendencia que supone el estudio de los tumores como masas heterogéneas en las que no solo la variabilidad entre las poblaciones tumorales es relevante, sino también la detección, caracterización y cuantificación de otros tipos celulares. Cabe destacar el papel de las células inmunes infiltradas, que en los últimos años se han establecido como elementos clave en la progresión del tumor, la eficacia de los tratamientos y, en última instancia, la supervivencia de los pacientes (Fridman et al., 2012; Galon et al., 2006; McGranahan y Swanton, 2015). Debido a su gran diversidad, se han reportado evidencias de que la presencia distintos tipos celulares puede dar lugar a diferentes efectos en la evolución del tumor, pudiendo favorecer a su detención o cooperar con las poblaciones neoplásicas produciendo un crecimiento más agresivo (Colbeck et al., 2017) (Figura 1.2). De esta forma y a nivel general, es posible encontrar tipos pro-tumorales como algunos linfocitos T CD4+ reguladores, macrófagos M2, células dendríticas o algunos tipos de linfocitos B (Bingle et al., 2002; Jahrsdörfer et al., 2010; Sarvaria et al., 2017); o tipos anti-tumorales como linfocitos T CD8+, linfocitos T CD4+ ayudantes, células *Natural Killer* y linfocitos B productores de inmunoglobulinas (Salgado et al., 2015). Además, es necesario añadir a este rol dicotómico del sistema inmune una dimensión más de complejidad, ya que la amplia variedad de relaciones que se establecen entre todos estos componentes produce que sea difícil extrapolar las consecuencias observadas en un tipo tumoral a otro (Colbeck et al., 2017; Fridman et al., 2012).

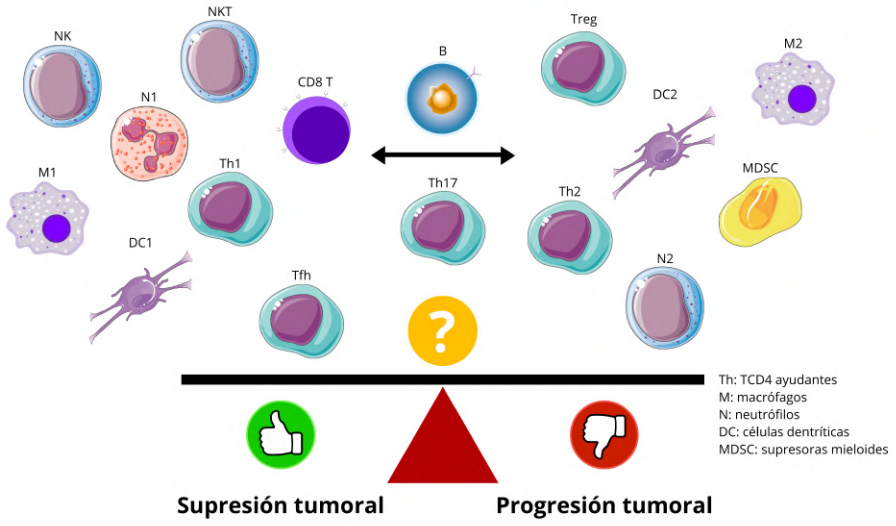

Figura 1.2: Papel dicotómico del sistema inmune en la progresión tumoral. Figura adaptada de Salgado et al. (2015).

La localización, densidad y organización funcional de estas células constituye lo que se conoce como el contexto inmune y, debido a su gran impacto clínico, en los últimos años han aparecido un gran número de nuevas terapias dirigidas a su modificación. Estos tratamientos consisten en quimioterapias inmunogénicas como el oxaliplatino (Fridman et al., 2012); anticuerpos inhibidores de puntos de control inmunitario como *Cytotoxic T lymphocyte-associated Antigen 4* (CTLA-4) o *Programmed Cell Death Protein 1* (PD-1), cuyo objetivo es aumentar la infiltración de células T CD8+ y, por tanto, la respuesta del sistema inmune en contra del tumor (Seidel et al., 2018); receptores co-estimuladores de la respuesta inmune como *Tumor Necrosis Factor Receptor Superfamily Member 18* (TNFRSF18) o *Tumor Necrosis Factor Receptor Superfamily Member 4* (TNFRSF4); o drogas anti-angiogénicas que decrecen los niveles de células mieloides supresoras o células T reguladoras, las cuales modulan la inmunidad anti-tumoral (Colbeck et al., 2017; Fridman et al., 2012).

1.1.2. Cáncer de mama

Entre los distintos subtipos de neoplasias, el cáncer de mama es el más común en mujeres en todo el mundo y es curable en aproximadamente el 70-80 % de los pacientes en estado no metastásico (Harbeck et al., 2019). Desde el punto de vista molecular, es una enfermedad altamente heterogénea, ya que las alteraciones que lideran la carcinogénesis son muy variadas, además de producirse cambios durante la progresión metastásica (Cejalvo et al., 2017). Clásicamente, se ha utilizado la clasificación propuesta por Perou y Sorlie conocida como subtipos intrínsecos del cáncer de mama (Perou et al., 2000; Sørlie et al., 2003). Está basada en los perfiles de expresión de 50 genes (PAM50) expresados por las células neoplásicas, aunque existen otras clasificaciones que incluyen otras variables como características histológicas o la expresión inmunohistoquímica de proteínas clave (Harbeck et al., 2019; Hon et al., 2016). Establece 4 subtipos principales:

- Luminal A: los tumores se caracterizan por la expresión de receptores de hormonas ER (receptor de estrógeno) y/o PR (receptor de progesterona) y la ausencia de HER2 (receptor del factor de crecimiento epidérmico 2).
- Luminal B: en estos casos, se produce la expresión de los receptores de hormonas junto con altos niveles de expresión de HER2.
- HER2 enriquecidos: estos tumores son principalmente negativos a receptores de hormonas

y presentan sobreexpresado HER2.

- Tumores basales: son aquellos con ausencia de expresión de receptores hormonales y de HER2. Por ello, son también denominados cáncer de mama triple negativo o TNBC.

Estas diferencias a nivel molecular dan lugar a enfermedades biológicamente diferentes con características propias (Figura 1.3). De hecho, debido a la alta infiltración de células inmunes que típicamente presentan, en los últimos años se han subdividido algunos de los subtipos mencionados teniendo en cuenta no solo la expresión de genes exclusivos de las células neoplásicas, sino también el perfil transcriptómico del contexto inmune. Por ejemplo, en el caso de los tumores HER2+ enriquecidos y TNBC, debido a que presentan un mayor número de TILs, han sido subdivididos en 3 y 6 subclústeres, respectivamente (Kroemer et al., 2015).

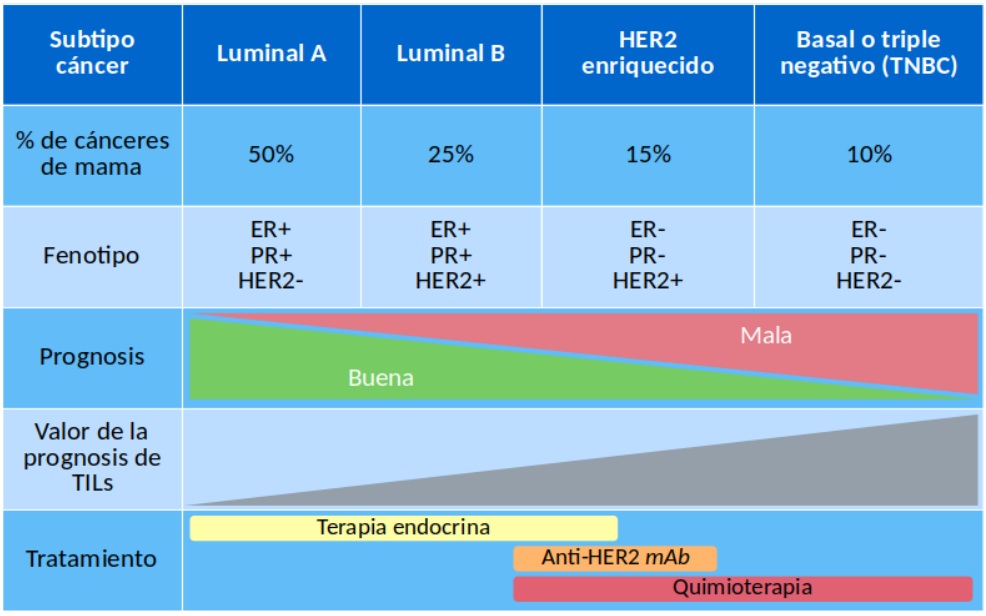

Figura 1.3: Comparativa de los subtipos intrínsecos del cáncer de mama. Figura adaptada de `www.pathophys.org/breast-cancer/`.

Respecto a los efectos de las células inmunes infiltradas en la prognosis de los pacientes, se ha demostrado que el análisis de la abundancia relativa de los tipos celulares en diferentes subtipos de tumor permite llevar a cabo predicciones más precisas sobre la evolución de la enfermedad. Concretamente, la presencia de linfocitos T CD8+ está fuertemente correlacionada con la supervivencia de los pacientes, mientras que la presencia de linfocitos T CD4+ reguladores se ha relacionado tanto con buenos como con malos resultados (Salgado et al., 2015). Por otra parte, los macrófagos han sido clásicamente relacionados con un efecto inmunosupresor (Bingle et al., 2002), produciendo tasas mayores de crecimiento e invasividad en casos metastásicos (Jeong et al., 2019), una mayor resistencia a tratamientos e inhibiendo a las células efectoras como los linfocitos T CD8+ (Cohen y Blasberg, 2017; Qiu et al., 2018). Además, los efectos de los tratamientos también varían en función de la composición del contexto inmune. Por ejemplo, en tumores HER2 enriquecido, se ha demostrado que la presencia de TILs es beneficioso en pacientes tratados con trastuzumab, un anticuerpo monoclonal dirigido contra HER2. También se han encontrado beneficios con el uso de quimioterapias neoadyuvantes en caso de que exista respuesta inmune contra el tumor, llegando incluso a incrementar la infiltración de células inmunes anti-tumorales (Moo et al., 2018).

1.1.3. Estudio del contexto inmune: *single-cell RNA-seq*

Debido al gran número de evidencias reportadas sobre el efecto de las células inmunes en la evolución y los tratamientos en cáncer de mama, es imprescindible conocer los tipos celulares presentes con el objetivo de conseguir terapias más especializados y alcanzar la meta de la medicina personalizada (McGranahan y Swanton, 2015). Tradicionalmente, la identificación de las poblaciones de células inmunes se ha realizado a nivel de proteína mediante la combinación de técnicas inmunohistoquímicas, inmunofluorescencia y citometría de flujo, seguido del análisis manual de las imágenes o mediante complejos algoritmos. Estas aproximaciones no solo son muy precisas, sino que permiten conocer la localización espacial de las células en el tumor. Sin embargo, no permiten el análisis de grandes cantidades de datos y, por lo tanto, pierden precisión en la caracterización del estatus funcional de las células, aspecto fundamental tal y como se ha discutido durante las secciones anteriores. Generalmente, se basan en pequeñas combinaciones de marcadores génicos preseleccionados, de forma que el número de tipos celulares que pueden ser estudiados simultáneamente es limitado (Newman et al., 2019).

Es en este punto donde las tecnologías de Secuenciación de Nueva Generación o NGS basadas en transcriptoma (*RNA-seq*) han emergido como una prometedora herramienta con posible utilidad clínica en el campo del cáncer. Se trata de una tecnología barata, reproducible, robusta y escalable que produce datos cuantitativos en lugar de las tecnologías más cualitativas utilizadas en proteómica, siendo capaz de capturar el perfil de expresión de un gran número de genes a la vez. Todas estas características la convierten en una herramienta ideal para su uso en medicina de precisión (Collins y Varmus, 2015), ya que suple las limitaciones de las aproximaciones tradicionales. Dentro de esta técnica, existen dos variantes principales que se diferencian en el nivel al que son tomados los datos de expresión génica:

- *Bulk RNA-seq*: los niveles de expresión corresponden a nivel de tejido y, por tanto, son el resultado del sumatorio de los diferentes tipos celulares presentes en la muestra.
- *Single-cell RNA-seq* (*scRNA-seq*): los datos son obtenidos de cada una de las células presentes en la muestra. Esto da lugar a un nivel de resolución mayor, ya que es posible conocer el estatus transcriptómico de cada célula individual.

Es esta última variante la que en los últimos años está revolucionando el campo de la Biomedicina, ya que permite explorar la heterogeneidad presente en los tejidos comentada en las secciones anteriores y, respecto al caso concreto de los tumores y el cáncer de mama, determinar las diferentes poblaciones celulares presentes en el micro-entorno tumoral, aumentando así el entendimiento sobre su papel en la enfermedad (Chung et al., 2017; Savas et al., 2018). Además, en última instancia, la aplicación translacional del *scRNA-seq* tiene la capacidad de mejorar el diagnóstico, el pronóstico, la selección de terapias dirigidas y la monitorización no invasiva de los pacientes (Navin, 2015).

Sin embargo, a pesar de la gran proyección que presenta, es necesario tener en cuenta que también tiene una serie de desventajas. Se trata de una metodología relativamente cara, haciendo que, por el momento, no sea práctico el análisis de grandes cohortes de muestras e impidiendo su aplicación generalizada en el ámbito clínico (Avila Cobos et al., 2018). Además, existen ciertas limitaciones inherentes a la técnica, como una baja eficiencia de captura de ARN y unos niveles más altos de ruido técnico en comparación con *bulk RNA-seq* (Chen et al., 2019). Respecto a naturaleza de los datos generados, debido a que se trata de una tecnología joven, aún no existen estándares a la hora de llevar a cabo los procesos de normalización y clusterización. Los métodos utilizados tradicionalmente en *bulk RNA-seq* no están pensados para matrices de expresión con un número tan alto de muestras y con una gran cantidad de ceros (matrices dispersas), particularidad debida a la baja cantidad de ARN de partida. Estas

propiedades suponen grandes retos que deben ser superados, pero también hacen que los análisis *bulk RNA-seq* aún sean el estándar utilizado y representen la mayor parte de los datos públicos de *RNA-seq* disponibles actualmente a pesar de que ignoran la proporción de tipos celulares que están midiendo (Stegle et al., 2015). De hecho, este último aspecto debe ser tenido en cuenta con especial relevancia en casos de tumores con una gran proporción de células infiltradas, ya que en análisis de expresión diferencial comparando condiciones puede suponer un importante factor de confusión (Avila Cobos et al., 2018).

1.1.4. Métodos de deconvolución de datos *bulk RNA-seq*

De todo ello, surge la necesidad de métodos computacionales capaces de inferir la proporción de tipos celulares presentes en las muestras a partir de datos de expresión *bulk RNA-seq*. Herramientas que posibiliten esta tarea pueden ser utilizadas en varios sentidos. Por ejemplo, pueden servir como un posible sustituto de experimentos *scRNA-seq* debido a sus altos costes o a la imposibilidad de su aplicación, como es el caso de muestras de tejidos fijados con formalina o parafina (Newman et al., 2019). También pueden utilizarse como mecanismos que permitan controlar la contribución de cada tipo celular a los niveles de expresión tisular, evitando así posibles factores de confusión en análisis de expresión diferencial. Además, se vuelven imprescindibles en el caso del estudio de enfermedades como el cáncer de mama por su carácter inherentemente heterogéneo, tal y como se ha comentado anteriormente.

El proceso es conocido en el campo de las Matemáticas como deconvolución y consiste en la estimación de la señal individual de cada uno de los componentes, en este caso tipos celulares, a partir de una mezcla de los mismos, la muestra heterogénea. Actualmente, existen multitud de herramientas capaces de estimar la composición celular de muestras de tumor a partir de datos de expresión *bulk RNA-seq*, como CIBERSORT (Newman et al., 2019), TIMER (Li et al., 2017) o EnrichR (Kuleshov et al., 2016), entre otras. El objetivo de todas ellas es identificar en qué medida el transcriptoma de cada tipo celular contribuye en el transcriptoma tumoral medido en la muestra.

El marco de trabajo de la deconvolución de datos *bulk RNA-seq* parte de que la expresión de un determinado gen en una muestra heterogénea puede ser modelada como una suma ponderada, es decir, una combinación linear, de los valores de expresión de cada tipo celular presente en la muestra. Para ello, es necesaria la asunción de que cada tipo celular tiene niveles de expresión similares entre diferentes muestras. Por lo tanto, el problema se puede reformular algebraicamente de la siguiente manera (Ecuación 1.1):

$$T_{ij} = \sum_{k=1}^{K} C_{ik} \cdot P_{kj} + e_{ij}; \quad i = 1 \dots M \quad \text{y} \quad j = 1 \dots N \tag{1.1}$$

donde T_{ij} es el valor de expresión del gen i obtenido de la muestra tisular j; C_{ik} es el nivel de expresión medio del gen i en el tipo celular k; P_{kj} es la proporción del tipo celular k en la muestra j; e_{ij} es un término de error no medible; K es el número de tipos celulares; M el número de genes; y N el número de muestras. En la Figura 1.4, se muestra un esquema representando gráficamente el marco de trabajo utilizado en deconvolución con la misma notación utilizada en la Ecuación 1.1 en forma matricial.

Basándose en esta formulación, algunos algoritmos tratan de obtener la proporción de TILs mediante el análisis de sets de enriquecimiento de marcadores específicos de cada tipo celular (Kuleshov et al., 2016). Otros, utilizan diferentes modelos matemáticos para deconvolucionar los niveles de expresión de la muestra total en los niveles de expresión de las distintas poblaciones

celulares. Ambas aproximaciones necesitan como entrada un set de marcadores predefinidos o los perfiles transcriptómicos completos de los tipos celulares para su uso como referencia.

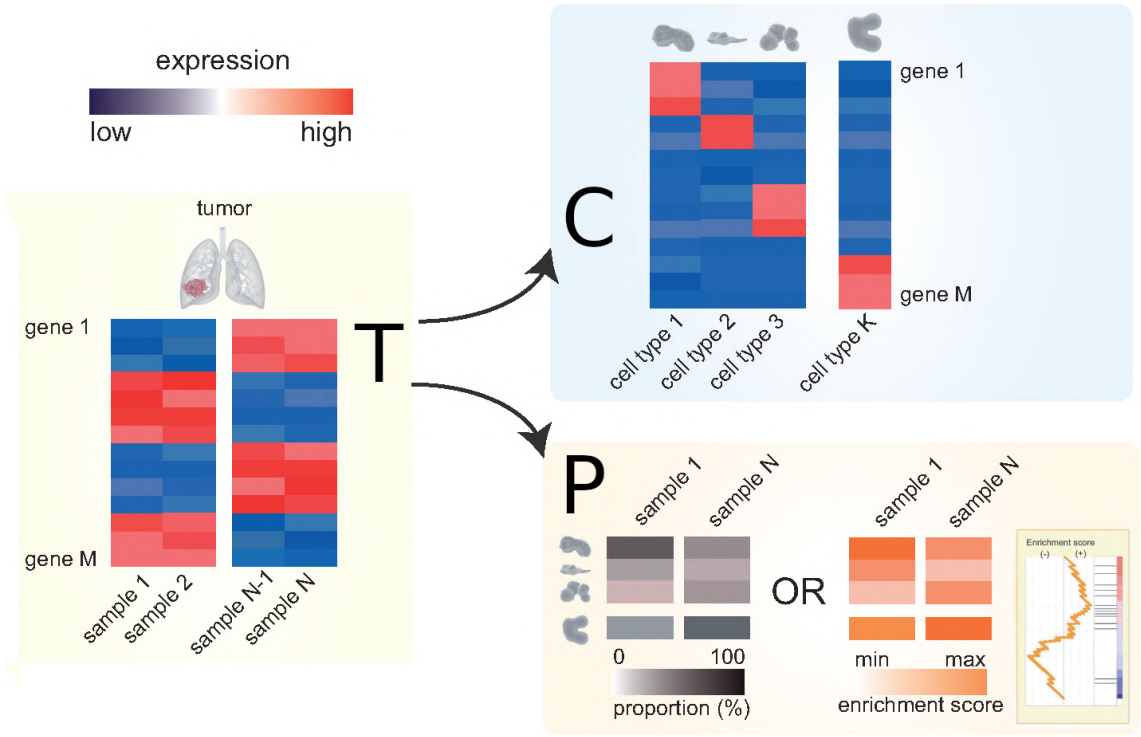

Figura 1.4: Esquema de los elementos principales del problema de deconvolución: T consiste en la matriz de expresión de las muestras de tumor con M genes y N muestras (datos observados); C es la matrix de expresión media para cada tipo celular con M genes y K tipos celulares; P es la matriz que contiene las proporciones de cada tipo celular en cada muestra (composición relativa) con K filas y N columnas. Figura modificada de Avila Cobos et al. (2018).

Dado que el fin último del proceso es encontrar la solución al sistema de ecuaciones mostrado en la Ecuación 1.1, dentro del rango de herramientas que utilizan modelos matemáticos para su resolución, típicamente se han usado algoritmos como mínimos cuadrados ordinarios con el fin de minimmizar las diferencias entre $C \cdot P$ y los valores observados P; o regresión de soporte vectorial con kernel linear (Newman et al., 2019). Sin embargo, hasta la fecha de publicación de digitalDLSorter (Torroja y Sanchez-Cabo, 2019), modelo sobre el que gira el presente Trabajo Fin de Máster, aún no existían métodos que implementaran el uso de Redes Neuronales Profundas (DNN), algoritmos de Aprendizaje Automático que están revolucionando dicha área debido a los impresionantes resultados que ofrecen (Sejnowski, 2020). Además, la mayoría de métodos hasta la fecha presentan el problema de que las referencias que ofrecen para su uso en el contexto de la cuantificación de células inmunes infiltradas consisten en perfiles transcriptómicos de células mononucleares de sangre periférica (PBMCs) procedentes de las bases de datos públicas. Este hecho produce que la precisión de estas herramientas se vea limitada, ya que las células presentes en el micro-entorno tumoral cambian significativamente sus perfiles transcriptómicos dependiendo del tejido y del contexto en el que se encuentran (Berglund et al., 2018; Cohen y Blasberg, 2017; Seo et al., 2018).

1.1.5. digitalDLSorter: método de deconvolución basado en Redes Neuronales Profundas

Con el fin de superar estos problemas, nace digitalDLSorter, un método basado en Aprendizaje Profundo ideado y construido por los co-tutores del presente proyecto para la enumeración

y la cuantificación de la composición de tipos celulares presentes en muestras *bulk RNA-seq*. Resumidamente, el modelo parte de datos *scRNA-seq* con los que lleva a cabo la generación de muestras *bulk* de composición celular conocida y, tras el entrenamiento de una DNN, ésta es capaz de inferir la proporción de cada tipo celular presente en las muestras. En el Capítulo 2, se desglosará con mayor profundidad el fundamento subyacente del modelo y cada uno de los pasos llevados a cabo. Sin embargo, a pesar de los buenos resultados presentados por Torroja y Sanchez-Cabo (2019), la herramienta consiste en una *pipeline* en la que se implementan cada uno de los pasos secuencialmente mediante la llamada desde terminal de *scripts* escritos tanto en R (R Core Team, 2020) como en Python (Van Rossum y Drake, 2009). Los resultados generados en cada paso son escritos en disco en forma de ficheros tabulados, lo que realentiza el proceso y complica su uso por el usuario. Además, no ofrece la posibilidad del uso de modelos preentrenados con el fin de facilitar la tarea de deconvolución. Actualmente está disponible en GitHub (GitHub, 2018) donde se puede explorar el código fuente.

1.2. Planteamiento y justificación

Por las razones expuestas a lo largo de la sección anterior, en el presente Trabajo Fin de Máster se ha realizado la transformación de la *pipeline* original en un paquete de R denominado digitalDLSorteR. El objetivo consiste en llevar a cabo la unificación del código en un solo lenguaje y conseguir una mayor integración del modelo con el entorno típicamente utilizado para el análisis de datos *RNA-seq* (R/Bioconductor). Para mejorar la experiencia de usuario, es posible utilizar la herramienta de dos maneras: 1) utilizando modelos pre-entrenados en diferentes micro-entornos tumorales para la deconvolución directa de muestras *bulk RNA-seq* o 2) para la construcción de nuevos modelos partiendo de datos *scRNA-seq* propios. Respecto a la segunda forma de uso, la nueva implementación pasa de escribir en disco cada uno de los datos intermedios generados durante el proceso, a su almacenaje en un objeto de R, para lo cual se ha utilizado el sistema de R de programación orientada a objetos (POO) S4. Además, para permitir que usuarios con limitaciones respecto a *hardware* puedan utilizar la herramienta, el paquete ofrece la posibilidad del uso de ficheros HDF5 (*Hierarchical Data Format 5*) como *back-end* para evitar mantener objetos de gran tamaño cargados en memoria RAM *Hierarchical Data Format 5*. Estas y otras nuevas funcionalidades son comentadas en el Capítulo 2 con mayor detalle.

Asimismo, debido a la gran importancia del micro-entorno tumoral en cáncer, concretamente en cáncer de mama, se ha llevado a cabo la construcción de un modelo de deconvolución a partir de los datos procedentes del artículo Chung et al. (2017) mediante el uso de la herramienta. Para ello, en primer lugar, se han analizado los datos mediante la implementación de una *pipeline* basada en el paquete de R Seurat (Stuart et al., 2019) para, posteriormente, llevar a cabo la caracterización de las células mediante el uso del paquete de R SingleR (Aran et al., 2019) y el estudio manual de marcadores específicos de tipo celular (véase Capítulo 3). Estos datos, los cuales proceden directamente del contexto de los tumores para superar las limitaciones que presentan otros métodos de deconvolución comentadas en la Subsección 1.1.4, han sido utilizados como entrada en el paquete digitalDLSorteR. El objetivo es mostrar la aplicación de la herramienta y el alcance del modelo como método de deconvolución (véase Capítulo 4).

1.3. Objetivos y enfoque

En base al planteamiento comentado anteriormente, el presente Trabajo Fin de Máster se puede dividir en 3 objetivos principales:

1. Transformación de la *pipeline* en el paquete de R digitalDLSorteR (véase Capítulo 2).
2. Análisis de los datos *scRNA-seq* procedentes del artículo Chung et al. (2017) y caracterización de los tipos celulares presentes en ellos (véase Capítulo 3).
3. Construcción de un modelo para la cuantificación de células inmunes infiltradas presentes en el micro-entorno tumoral del cáncer de mama mediante el uso del paquete digitalDLSorteR a partir de los datos previamente analizados y caracterizados (véase Capítulo 4).

Estos objetivos principales, a su vez, pueden subdividirse en diferentes objetivos parciales necesarios para su consecución. Respecto a la construcción del paquete digitalDLSorteR:

- Diseño de las clases S4 cuyos objetos contendrán los datos generados durante la construcción de los modelos. Generación de las correspondientes funciones *getter* y *setter* en forma de funciones genéricas.
- Transformación de cada uno de los pasos de la *pipeline* original en funciones de R y migración del código en Python a R. Implementación de nuevas funcionalidades, como el uso de ficheros HDF5 como *back-end* o parámetros que permitan modificar aspectos del modelo.
- Implementación de funciones para el cálculo de métricas de error y la generación de gráficos para la evaluación de los modelos mediante el uso del paquete ggplot2 (v3.2.1) (Wickham, 2016b).
- Generación de la documentación correspondiente a cada una de las funciones y clases exportadas mediante el paquete roxygen2 (Wickham et al., 2020a).
- Generación de una viñeta ilustrativa con las principales funcionalidades. Integración de pequeños subconjuntos de datos con el fin de mostrar las principales funcioinalidades.
- Generación de los ficheros *DESCRIPTION* y *NAMESPACE*.

Respecto al análisis de los datos *scRNA-seq*:

- Descarga de las lecturas desde la base de datos *Sequence Read Archive* (SRA), alineamiento y cuantificación de los niveles de expresión.
- Manejo de los paquetes de R para el análisis de datos *scRNA-seq* scater (McCarthy et al., 2017) y Seurat. Preprocesamiento, normalización, búsqueda clústeres y visualización de las células.
- Caracterización de las células presentes en el conjunto de datos. Separación de células tumorales/no tumorales y uso de herramientas bioinformáticas y marcadores específicos para la caracterización de células no tumorales.

Finalmente, respecto a la construcción del modelo de deconvolución a partir de los datos caracterizados:

- Puesta en práctica de la herramienta desarrollada.
- Generación de un modelo de deconvolución para datos procedentes de cáncer de mama a partir de las células caracterizadas anteriormente.
- Comparativa de diferentes modelos generados con diferentes parámetros.
- Aplicación del modelo entrenado sobre un conjunto de datos *bulk RNA-seq* reales.

1.4. Materiales utilizados

Implementación del paquete digitalDLSorteR

Dado que el paquete está construido completamente en R (v3.6.0), se decidió utilizar para su implementación el IDE RStudio (v1.3.1056) (RStudio Team, 2016). Además, se hizo uso de los paquetes devtools (v2.2.2) (Wickham et al., 2020b) y usethis (v1.5.1) (Wickham y Bryan, 2020), herramientas orientadas a facilitar el desarrollo de paquetes en R. La documentación, así como el fichero NAMESPACE, fueron generados mediante el empleo del paquete roxygen2 (v7.1.1) (Wickham et al., 2020a). Respecto a los paquetes utilizados para la implementación de partes concretas del paquete, son detallados en el Capítulo 2. El control de versiones se realizó utilizando GitHub desde terminal.

Análisis de datos *scRNA-seq*

Las tareas relacionadas con la modificación de los ficheros gtf de las anotaciones y fasta del genoma de referencia fueron llevadas a cabo mediante la implementación de *scripts* en bash y en awk. La descompresión de las lecturas del artículo Chung et al. (2017) descargadas en formato sra desde la base de datos SRA (SRP066982) fue llevada a cabo mediante el comando `fastqdump` de la herramienta sratools (v2.10.5). El filtrado y preprocesamiento de las lecturas se realizó mediante la herramienta cutadapt (v1.9) (Martin, 2011). La medición de las métricas de calidad de las lecturas en cada paso del preprocesamiento se llevó a cabo mediante el programa fastqc (v0.11.8.) (Andrews et al., 2012). El alineamiento de las lecturas preprocesadas se llevó a cabo mediante STAR (v2.3.1) (Dobin y Gingeras, 2015), mientras que la cuantificación de los tránscritos con RSEM (v1.3.2) (Li y Dewey, 2011). Para más detalles sobre los parámetros utilizados, véase la Sección 3.3. Tras la obtención de las matrices de expresión, el análisis completo de los datos *scRNA-seq* se desarrolló bajo el entorno de R utilizando el IDE RStudio. Los paquetes principalmente utilizados para el análisis de los datos fueron scater (v1.14.6), Seurat (v3.1.3) y SingleR (v1.0.5). Respecto al paso de normalización, se utilizó el paquete sctransform (v0.2.1) (Hafemeister y Satija, 2019) desarrollado por los autores de Seurat.

Aplicación del paquete

La construcción de los modelos de deconvolución mediante el uso de la herramienta implementada fue llevada a cabo mediante el uso de R en el IDE RStudio. El análisis de correlación de las proporciones obtenidas sobre los datos reales *bulk RNA-seq* fue llevado a cabo mediante el paquete de R GGally (v1.4.0) (Schloerke et al., 2020).

Hardware utilizado

Todas las tareas relativas al tratamiento de las lecturas 'crudas' del artículo (Chung et al., 2017), exceptuando la limpieza de los ficheros gtf y fasta, fueron ejecutadas en el clúster de computación de la Unidad de Bioinformática del CNIC con acceso remoto vía ssh y usando el gestor de colas *Son Of Grid Engine* (antiguamente *Sun Grid Engine*) con comandos qsub. El resto de los análisis y todo lo relacionado con el desarrollo del paquete, se ha realizado en un equipo Intel(R) Core(TM) i5-6500 con 32GB (*gigabytes*) de RAM bajo el sistema operativo CentOS 7.1611.

2

Desarrollo del paquete digitalDLSorteR

2.1. Introducción

Tal y como se ha comentado en la sección Objetivos y enfoque del capítulo anterior, el primer objetivo principal del proyecto consistió en la transformación de la *pipeline* digitalDLSorter en un paquete de R. Los *scripts* originales que la componen fueron implementados como funciones formales que trabajan sobre una serie de objetos definidos mediante el sistema de clases S4. A lo largo de este capítulo, se hará una exposición general del fundamento del modelo, el diseño de las clases y objetos, el flujo de trabajo que presenta el paquete y las nuevas funcionalidades que ofrece. Respecto a los recursos empleados para su implementación, pueden encontrarse en la sección Materiales utilizados.

2.2. Perspectiva general y fundamento del modelo

El modelo ideado e implementado por Torroja y Sanchez-Cabo (2019) consiste en un método basado en Aprendizaje Profundo para la cuantificación de las proporciones de tipos celulares presentes en muestras *bulk RNA-seq*, funcionando, además, como clasificador de perfiles *single-cell*. El método consta de 3 pasos principales en los que, a partir de datos *scRNA-seq* previamente caracterizados, se construye una Red Neuronal Profunda capaz de inferir las proporciones de tipos celulares presentes en muestras *bulk RNA-seq* que provengan de un mismo contexto (Figura 2.1):

1. **Simulación de perfiles *single-cell* (si es necesario):** el modelo comienza con la matriz de datos *scRNA-seq*. En el caso de que haya tipos celulares representados por un número de perfiles transcriptómicos menor que un determinado umbral o el número de células de partida sea demasiado bajo, son simulados nuevos perfiles mediante el uso del modelo ZINB-WaVE (Risso et al., 2018). Esta situación de subrepresentación de determinados tipos celulares es típica en muestras *scRNA-seq* procedentes de tumor, ya que suele haber una mayoría de células tumorales, mientras que la porción restante está compuesta por multitud de tipos celulares.

2. **Simulación de perfiles *bulk RNA-seq*:** tras la obtención de los perfiles *scRNA-seq*

finales, las células son separadas en un conjunto de datos de entrenamiento (65 %) y de test (35 %) y son usadas para la generación de muestras *bulk RNA-seq* de composición celular conocida.

3. **Entrenamiento y evaluación de una Red Neuronal Profunda:** los perfiles transcriptómicos del conjunto de datos de entrenamiento son utilizados para el entrenamiento de una Red Neuronal Profunda. Tras el proceso, el desempeño del modelo es evaluado sobre el conjunto de datos de test, obteniéndose una serie de métricas con las que poder determinar su precisión.

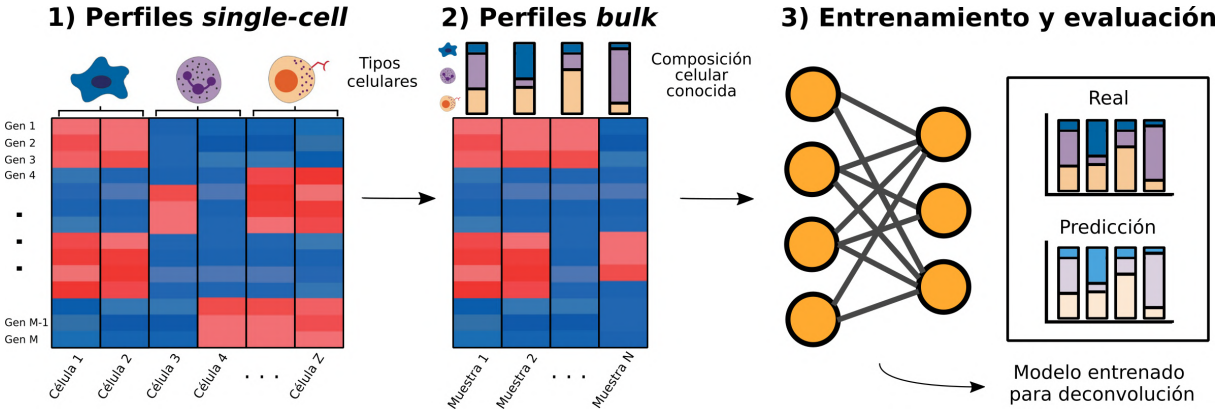

Figura 2.1: Perspectiva general del modelo. 1) Perfiles *single-cell* con células caracterizadas. 2) Simulación de perfiles *bulk* con composición celular conocida. 3) Entrenamiento y evaluación de la Red Neuronal Profunda.

Teniendo en cuenta el marco de trabajo del problema de deconvolución de datos de expresión génica expuesto en la Subsección 1.1.4 (Figura 1.4), el fundamento del método digitalDLSorter consiste en el entrenamiento de una red neuronal con perfiles *bulk RNA-seq* sintéticos (matriz T_{ij} conocida) de composición celular conocida (matriz P_{kj} conocida) construidos a partir de perfiles *single-cell* conocidos (matriz C_{ijk} conocida). Las muestras *bulk RNA-seq* son generadas precisamente como una suma ponderada de perfiles *single-cell* en función de las proporciones definidas en P_{kj} (véase Sección 2.4), por lo que el modelo, tras su entrenamiento, es capaz de inferir la proporción de nuevas muestras *bulk RNA-seq* cuyo contexto sea similar al de los perfiles *single-cell* de partida. La red neuronal aprende a estimar la composición de tipos celulares mediante la búsqueda de patrones en los perfiles de expresión que los identifiquen. Considerando los impresionantes resultados obtenidos por este tipo de algoritmos (Sejnowski, 2020), el resultado es un método de deconvolución preciso y fácil de utilizar mediante el uso de modelos preentrenados.

A continuación, se explican las principales características de la implementación del modelo como paquete de R, yendo desde el sistema de clases que utiliza hasta el flujo de trabajo requerido para su uso.

2.3. Sistema de clases y objetos

Uno de los principales objetivos planteados respecto a la construcción del paquete fue evitar la lectura/escritura constante de ficheros en disco implementada en la *pipeline* original con el fin de acelerar los tiempos de ejecución y hacer más cómodo su uso por el usuario. Por ello, se llevó a cabo la implementación de una serie de clases utilizando el sistema de POO S4 con el fin de construir objetos que sirvan de 'contenedor' de los diferentes datos generados a lo largo

del proceso. Se eligió utilizar este sistema no solo por su inmensa presencia entre los paquetes publicados en Bioconductor (en la versión 3.3 existen 3.158 clases definidas en 609 paquetes), sino por presentar tanto la flexibilidad que ofrece el hecho de que los métodos y las clases operen de forma independiente propia del sistema S3, como una definición más formal de las clases que este último. Esto permite construir ricas y complicadas representaciones de datos que puedan parecer sencillas para el usuario final y llevar a cabo comprobaciones de validación sobre cómo está construido el objeto y los datos que contiene. De esta forma, se garantiza la validez de los objetos y se evitan posibles errores de ejecución por inconsistencias internas (Hansen, 2015; Wickham, 2016a). Estas características hacen de las clases S4 el sistema de POO ideal para el paquete digitalDLSorteR, ya que:

- El hecho de poder comprobar la validez de los objetos permite asegurar la integridad de los datos en cada paso del modelo.
- La posibilidad de construir complejas estructuras en las que almacenar datos de diferente naturaleza en diferentes *slots* es ideal para el planteamiento seguido por el modelo, ya que se generan diferentes datos intermedios que deben ser accesibles en diferentes puntos del proceso.
- Este hecho permite unificar toda esta información en un único objeto. Para el usuario medio, trabajar sobre un único objeto resultará más sencillo, mientras que los usuarios avanzados podrán acceder y modificar los datos que contiene con total transparencia gracias a la implementación de métodos *getter* y *setter*.

El paquete presenta una clase principal (*DigitalDLSorter*) que sirve como núcleo sobre el que todas las funciones acceden y almacenan los datos producidos. Además, se han implementado otras dos clases (*ProbMatrixCellTypes* y *DigitalDLSorterDNN*) con el fin de almacenar datos relacionados entre sí de forma unificada.

2.3.1. Clase *DigitalDLSorter*

Se trata de la clase principal del paquete. Sus instancias se encargan de contener los diferentes datos generados a lo largo de la construcción del modelo de deconvolución. Está compuesta por 11 *slots*, dentro de los cuales se hace uso tanto de clases S3 como de clases S4, algunas procedentes del entorno de Bioconductor y otras implementadas *de novo* en el paquete. Las clases externas utilizadas, junto a una breve descripción de cada una, son:

- *SingleCellExperiment*: se trata de una clase ofrecida por el equipo de Bioconductor mediante su paquete homónimo (Lun y Risso, 2019) que funciona como contenedor para el almacenamiento y la manipulación de datos genómicos *single-cell*. Se decidió utilizar para el mantenimiento de los datos *scRNA-seq* en digitalDSorteR debido a que es la estructura de datos utilizada por la mayoría de paquetes de Bioconductor para este tipo de análisis. Además, permite la representación de las matrices de expresión en forma de matrices dispersas utilizando la clase *dgCMatrix* del paquete Matrix (Bates y Maechler, 2019), proporcionando así un menor consumo de memoria RAM. Esto es posible gracias al hecho de que los datos *scRNA-seq* presentan una gran cantidad de ceros, tal y como se ha comentado en la Subsección 1.1.3.
- *ZINBParams*: como se ha comentado anteriormente, el modelo lleva a cabo la simulación de nuevos perfiles *single-cell* en el caso de que haya tipos celulares subrepresentados. Para ello, primeramente se realiza la estimación de los parámetros del modelo ZINB-WaVE a partir de los datos *scRNA-seq* reales mediante el uso del paquete splatter (Zappia et al.,

2017). El objeto generado tras el proceso es de la clase *ZINBParams* y contiene toda la información necesaria para la posterior simulación de los perfiles *single-cell* sintéticos.

- *SummarizedExperiment*: es otra clase ofrecida por el equipo de Bioconductor en el paquete homónimo (Morgan et al., 2019) para el almacenamiento de datos *bulk RNA-seq*. La clase *SingleCellExperiment* hereda de ésta, por lo que el diseño que presentan es muy similar. Cabe destacar que permite el almacenamiento de las matrices de expresión como objetos *HDF5Matrix*, clase del paquete HDF5Array (Pagès, 2020) que permite el acceso desde R de grandes cantidades de datos almacenados en ficheros HDF5 sin la necesidad de mantenerlos cargados en memoria. En la Sección 2.4, se comenta con mayor detalle su funcionamiento y las razones de su uso en digitalDLSorteR.

Respecto a los *slots*, a continuación, se comenta una breve descripción de los objetos y la información que contienen:

- *single.cell.real*: contiene un objeto de la clase *SingleCellExperiment* con los datos *scRNA-seq* originales a partir de los cuales se construirá el modelo.

- *zinb.params*: contiene un objeto de la clase *ZINBParams* con los parámetros estimados a partir de los perfiles transcriptómicos reales.

- *single.cell.final*: contiene un objeto de la clase *SingleCellExperiment* en el que se encuentran tanto los datos *scRNA-seq* originales como los simulados. En el caso de que no se lleve a cabo la simulación de nuevos perfiles *single-cell*, contendrá únicamente los perfiles reales.

- *prob.cell.types*: contiene una lista con dos elementos, *train* y *test*. Cada elemento es un objeto de la clase *ProbMatrixCellTypes* con toda la información relacionada con las matrices de composición celular generadas para la simulación de los perfiles *bulk RNA-seq*.

- *bulk.sim*: contiene una lista con dos elementos, *train* y *test*. Cada elemento es un objeto de la clase *SummarizedExperiment* que contiene los perfiles *bulk* simulados mediante la agregación de los perfiles *single-cell*.

- *final.data*: de nuevo, es una lista con dos elementos, *train* y *test*. Cada uno contiene los datos finales que serán utilizados para el entrenamiento de la red neuronal (*train*) y para la evaluación de la misma (*test*).

- *trained.model*: contiene un objeto de la clase *DigitalDLSorterDNN*. En él, se encuentra toda la información relativa a la Red Neuronal Profunda: el modelo entrenado, los resultados sobre el conjunto de datos de test, las métricas de evaluación, etc.

- *deconv.data*: lista en las que se pueden cargar datos *bulk RNA-seq* nuevos a deconvolucionar.

- *deconv.results*: lista en la que se almacena la matriz de composición celular obtenida tras llevar a cabo la deconvolución de los perfiles *bulk* contenidos en *deconv.data*.

- *project*: cadena de caracteres indicando el nombre del proyecto.

- *version*: versión del paquete sobre el que fue construido el objeto en cuestión.

2.3.2. Clase *ProbMatrixCellTypes*

En los objetos de esta clase, se almacena toda la información relativa a las matrices de composición celular que determinarán la proporción de cada tipo celular en cada muestra *bulk RNA-seq* simulada. Los *slots* que contiene son:

- *prob.matrix*: contiene la matriz de proporción de tipos celulares (P_{jk}). Las filas corresponden con las muestras *bulk* que serán generadas (j), mientras que las columnas son cada uno de los tipos celulares presentes en los datos *single-cell* iniciales (k). Cada entrada corresponderá a la proporción del tipo celular k en la muestra j, lo que, en última instancia, se traducirá en el número de células de cada tipo celular k que compondrán esa muestra *bulk*.
- *cell.names*: contiene una matriz en la que se especifican los nombres de cada una de las células que compondrán cada muestra *bulk*. Estas células se muestrean aleatoriamente en función del tipo celular al que pertenecen.
- *set.list*: lista de células segregadas en función del tipo celular al que pertenezcan.
- *set*: vector con el nombre de las células presentes en el experimento.
- *plots*: lista en la que se almacenan gráficos que representan la distribución de los tipos celulares en la matriz de composición celular. Dado que las proporciones se generan mediante 5 métodos distintos con el fin de evitar sesgos (véase Sección 2.4), es posible observar la distribución de las probabilidades generadas por cada método.
- *type.data*: cadena de caracteres indicando si el conjunto de proporciones y las células que compondrán las muestras corresponden a los datos de entrenamiento o a los de test.

En la Sección 2.4, se comentan con detalle los métodos implementados para la construcción de las matrices de composición celular, así como la simulación de las muestras *bulk*.

2.3.3. Clase *DigitalDLSorterDNN*

Los objetos de esta clase contienen toda la información relativa a la Red Neuronal Profunda. Aquellos pasos que tienen que ver con su uso han sido implementados mediante el paquete de R Keras (v2.3.0) (Allaire y Chollet, 2020), una interfaz de Python para R en la que se utiliza en segundo plano el módulo de Python del mismo nombre. Éste módulo, a su vez, funciona como interfaz para el *back-end* que se seleccione, siendo, en este caso, Tensorflow (v2.2.0) (Abadi et al., 2015). Por ello, en algunos *slots* de esta clase son almacenados objetos de las clases que implementa Keras, las cuales forman parte del sistema de clases R6 o de referencia. Este sistema está basado en el paradigma de POO del encapsulamiento, que es el utilizado por otros lenguajes de programación como Python o Java (Wickham, 2016a). La razón es que, dado que los pasos relacionados con la red neuronal son ejecutados por Python y el *back-end* de Tensorflow, estos objetos son, en última instancia, estructuras no nativas de R, sino de Python. Los *slots* son:

- *model*: contiene la red neuronal entrenada con la que se podrán llevar a cabo predicciones de nuevas muestras. El *slot* puede ser un objeto R6 *keras.engine.sequential.Sequential*, el cual contiene la arquitectura de la red, los pesos obtenidos tras el entrenamiento y el estado del optimizador tras la última época del entrenamiento. También puede ser una lista que contiene la arquitectura y los pesos en objetos nativos de R para posibilitar el almacenamiento en disco de modelos entrenados en ficheros rda o rds. Esto se debe a que este tipo de ficheros únicamente permiten el almacenamiento de objetos nativos

de R, no estructuras complejas de datos procedentes de otros lenguajes. A pesar de la pérdida del estado del optimizador, se mantiene la información necesaria para hacer nuevas predicciones.

- *training.history*: contiene la progresión de las métricas de evaluación seleccionadas durante el entrenamiento.
- *eval.stats.model*: contiene los resultados de las métricas de evaluación del modelo sobre los datos de test.
- *predic.results*: contiene la matriz de composición celular predicha para cada una de las muestras que componen el conjunto de datos de test.
- *cell.types*: vector de tipos celulares incluidos en el modelo.
- *features*: vector de genes tenidos en cuenta por el modelo.
- *eval.stats.samples*: contiene las métricas de error calculadas comparando los resultados predichos con los resultados reales sobre el conjunto de datos de test.

2.4. Flujo de trabajo y funciones principales

digitalDLSorteR se ha construido con el fin de que sirva tanto como una herramienta de deconvolución mediante el uso de modelos preentrenados integrados en el paquete (Figura 2.2.a), como para permitir la posibilidad de construir nuevos modelos a partir de datos *scRNA-seq* de diferentes entornos cuyas células estén previamente caracterizadas (Figura 2.2.b). Es por ello que el paquete ofrece estos dos posibles flujos de trabajo. A continuación, se expondrá cada uno añadiendo ejemplos de código incluidos en la documentación del paquete. Algunos de estos ejemplos pueden ejecutarse tras su intalación gracias a que los datos utilizados se encuentran en el directorio digitalDLSorteR/data. Consisten en versiones reducidas de los datos originales con el fin de poder ejecutarse en ordenadores sin grandes recursos y en tiempos relativamente cortos. Además, muchos de los parámetros mostrados en los ejemplos han sido reducidos con el mismo fin (número de células y muestras *bulk* simuladas, etc.). Se comentarán algunas particularidades destacables respecto a la implementación de las funciones, aunque es importante tener en cuenta que únicamente serán expuestos aspectos relevantes y aquellas funcionalidades vitales para el paquete. No se han explicado muchos de los parámetros disponibles en las funciones ni algunas funciones secundarias. Para más información, es posible visitar tanto la documentación del paquete como la viñeta en la que se muestra el código y la salida generada en cada uno de los pasos. Toda esta información, así como las instrucciones para su instalación, están comentados en el Material Suplementario A.

2.4.1. Uso de modelos preentrenados

Con el objetivo de facilitar el uso del paquete como herramienta de deconvolución, es posible el uso de modelos preentrenados integrados en él. Por el momento, el paquete únicamente ofrece los modelos construidos en el presente proyecto a partir de los datos del artículo Chung et al. (2017), por lo que solo es posible deconvolucionar muestras de cáncer de mama. Próximamente, se añadirán nuevos modelos para la deconvolución de datos procedentes de otros entornos, como cáncer de colon o PBMCs. Los modelos son almacenados en el directorio digitalDLSorteR/data en forma de ficheros rda como objetos *DigitalDLSorterDNN*, tal y como es recomendado (Wickham, 2015). En éstos, el *slot model* contiene una lista con la arquitectura de la red y los

pesos obtenidos tras el entrenamiento, que es la información mínima necesaria para su uso como predictor.

Respecto a su utilización, basta con cargar en R la matriz de datos *bulk RNA-seq* que se desea deconvolucionar, de forma que los genes se encuentren en las filas y las muestras en las columnas. De esta manera, el usuario no tiene por qué utilizar objetos implementados en el paquete, facilitando el proceso. La función `deconvDigitalDLSorter` recibe como argumento dicha matriz y la elección del modelo que se quiere utilizar, siendo, por el momento, únicamente seleccionables los modelos `'breast.chung.generic'` y `'breast.chung.specific'` (véase Capítulo 4). Los datos son normalizados por defecto y, tras su ejecución, devuelve la matriz de composición celular resultante. En el caso de que haya tipos celulares altamente similares que se quieran colapsar bajo una misma etiqueta, se han implementado dos posibles argumentos: `simplify.set` y `simplify.majority`. Permiten la especificación en forma de lista de uno o varios grupos de tipos celulares que se quieren comprimir. Se diferencian en que el primer argumento establece una nueva etiqueta provista por el usuario, mientras que, el segundo, asigna la suma de las proporciones al tipo celular con mayor proporción en cada muestra. Finalmente, es posible observar los resultados en forma de gráfico de barras mediante la función `barPlotCellTypes`. En la Code Box 2.1, se muestra un ejemplo de este modo de uso sobre un subconjunto de los datos originales *bulk RNA-seq* de cáncer de mama del proyecto TCGA (*The Cancer Genome Atlas*) (Koboldt et al., 2012; Ciriello et al., 2015).

```
deconvResults <- deconvDigitalDLSorter(
  data = TCGA.breast.small,
  model = "breast.generic",
  normalize = TRUE
)
## barplot showing results
barPlotCellTypes(deconvResults)
```

Code Box 2.1: Uso de la función `deconvDigitalDLSorter`. El objeto `TCGA.breast.small` consiste en una matriz con genes en las filas y muestras en las columnas.

2.4.2. Construcción de nuevos modelos a partir de datos *scRNA-seq*

1. Carga de datos en un objeto *DigitalDLSorter*

digitalDLSorteR parte de datos *scRNA-seq* en los que las células han sido previamente caracterizadas. El primer paso consiste en la carga de éstos en un objeto *DigitalDLSorter*. Los datos pueden cargarse desde objetos *SingleCellExperiment* o desde ficheros tabulados/matrices dispersas almacenados en disco. En cualquier caso, deben proporcionarse tres piezas de información: la matriz de expresión génica; los metadatos de las células, en los que se debe incluir el tipo celular al que pertenecen; y los metadatos de los genes, en los que se debe incluir la notación utilizada en la matriz de expresión (symbol, etc.). En el caso de que los datos presenten tipos celulares subrepresentados o el número de células de partida sea particularmente bajo, deben ser cargados mediante la función `loadRealSCProfiles`, que los almacenará en el *slot single.cell.real* (Code Box 2.2). En caso contrario, la función `loadFinalSCProfiles` cargará los datos en el *slot single.cell.final* para ser utilizados directamente para la simulación de las muestras *bulk*. Es recomendable llevar a cabo el sobremuestreo de los perfiles *single-cell* con el fin de aumentar la variabilidad que presentarán las muestras *bulk* simuladas. Respecto a las unidades de cuantificación de la expresión génica utilizadas como entrada, es recomendable el uso de TPMs (tránscritos por millón) por ser una medida normalizada y relativamente estable. Esta propiedad se debe al hecho de que la suma de los TPMs de cada muestra es el mismo, permitiendo hacer comparaciones más sencillas (Wagner et al., 2012).

```r
DDLSChungSmall <- loadRealSCProfiles(
  single.cell.real = sc.chung.breast, ## SingleCellExperiment object
  cell.ID.column = "Cell_ID", gene.ID.column = "external_gene_name",
  min.cells = 1, min.counts = 1, project = "Chung_example"
)
```

Code Box 2.2: Ejemplo de la carga de datos *scRNA-seq* desde un objeto *SingleCellExperiment* en un objeto de la clase *DigitalDLSorter*.

2. Estimación de parámetros y simulación de nuevos perfiles *single-cell*

En el caso de que sea necesaria la simulación de nuevos perfiles *single-cell*, en primer lugar, se realiza la estimación de los parámetros que posteriormente serán utilizados para la tarea. Para ello, es utilizado el modelo ZINB-WaVE, un modelo binomial negativo cero-inflado diseñado para datos con alta dimensionalidad y con gran cantidad de ceros como los presentes en los experimentos *scRNA-seq*. Este modelo permite la especificación de diferentes covariables tanto para los genes como para las células, de forma que no solo ajusta la variabilidad presente en un determinado tipo celular, sino la del experimento completo. Los parámetros μ (nivel de expresión medio), θ (parámetro de dispersión alrededor de la media) y π (probabilidad de ceros) de la distribución negtiva binomial cero-inflada (ZINB) serán utilizados posteriormente para la simulación de nuevos perfiles *single-cell*. Este paso está implementado en el paquete mediante la función `estimateZinbwaveParams` (Code Box 2.3), la cual hace uso del paquete splatter (Zappia et al., 2017). Por defecto, siempre recibe como covariable el tipo celular, ya que es la señal principal que se quiere capturar. Pero, gracias a que ZINB-WaVE puede recibir otras covariables, mediante los argumentos `cell.cov.columns` y `gene.cov.columns` se puede especificar otra información de interés en función del conjunto de datos con el que se trabaje.

Después, mediante la función `simSingleCellProfiles`, los parámetros μ y θ son utilizados para la simulación de nuevos perfiles mediante el muestreo aleatorio de una distribución binomial negativa y el parámetro π para la introducción de ceros mediante el muestreo de una distribución binomial. Mediante el parámetro `n.cells`, es posible especificar el número de células por tipo celular que serán generadas (Code Box 2.3).

```r
## estimation of ZINB-WaVE parameters
DDLSChungSmall <- estimateZinbwaveParams(
  object = DDLSChungSmall,
  cell.ID.column = "Cell_ID",
  gene.ID.column = "external_gene_name",
  cell.type.column = "Cell_type",
  cell.cov.columns = "Patient",
  gene.cov.columns = "gene_length",
  threads = 2
)
## simulation of new profiles
DDLSChungSmall <- simSingleCellProfiles(
  object = DDLSChungSmall,
  cell.ID.column = "Cell_ID",
  cell.type.column = "Cell_type",
  n.cells = 10 # 1000 in real situations
)
```

Code Box 2.3: Ejemplo de la estimación de los parámetros del modelo ZINB-WaVE y su posterior uso para la simulación de nuevos perfiles *single-cell*.

3. Generación de las matrices de composición celular

Una vez se han obtenido los perfiles *single-cell* definitivos, se lleva a cabo la construcción de la matriz composición celular mediante la función `generateTrainAndTestBulkProbMatrix`. Determinará la proporción de los diferentes tipos celulares considerados en el modelo en cada una de las muestras *bulk* simuladas. Por lo tanto, en este paso también se decide el número de muestras *bulk* con el que se va a construir el modelo, siendo necesaria su fijación mediante el argumento `n.bulk.samples` (Code Box 2.4).

```
## data.frame with prior information about cell types
probMatrix <- data.frame(
  Cell_types = c("ER+", "HER2+", "ER+ and HER2+", "TNBC",
                 "Stromal", "Monocyte", "TCD4me", "BGC",
                 "Bmem", "DC", "Macrophage", "TCD8", "TCD4reg"),
  from = c(rep(30, 4), 1, rep(0, 8)),
  to = c(rep(70, 4), 50, rep(15, 8))
)
DDLSChungSmall <- generateTrainAndTestBulkProbMatrix(
  object = DDLSChungSmall,
  cell.type.column = "Cell_type",
  prob.design = probMatrix,
  n.bulk.samples = 200 # 30000 in real situations
)
```

Code Box 2.4: Ejemplo de la construcción de las matrices de composición celular. La función requiere a través del argumento `prob.design` un `data.frame` con información previa sobre los rangos en los que puede aparecer cada tipo celular. Esta información se puede inferir de la literatura o del experimento *scRNA.seq* en cuestión.

En relación al funcionamiento interno del proceso, en primer lugar, los datos *scRNA-seq* son separados en dos conjuntos, entrenamiento y test, de forma que las muestras *bulk* con las que se entrenará y validará la red neuronal serán simuladas a partir de células diferentes. De esta manera, se evita la mezcla de datos de entrenamiento y de test y, por tanto, el falseamiento de los resultados. Después, se generan las matrices de probabilidad que determinan la composición de las muestras *bulk* de entrenamiento y de test. Estas matrices presentarán el mismo número de filas que muestras serán simuladas (j) y el mismo número de columnas que tipos celulares hay en el experimento (k). Por tanto, no son más que la transpuesta de la matriz P_{kj} definida en la Subsección 1.1.4. Para evitar posibles sesgos debidos a la composición de las muestras *bulk*, éstas son generadas mediante distintas aproximaciones, dando lugar a 5 conjuntos de muestras con distribuciones diferentes:

1. En el primer conjunto, las proporciones celulares son muestreadas aleatoriamente de una distribución uniforme con límites predefinidos de acuerdo a unos rangos *a priori* proporcionados por el usuario mediante el argumento `prob.design`. Las muestras resultantes presentarán una composición similar a la información proporcionada.

2. El segundo conjunto es generado mediante la permutación aleatoria de los tipos celulares sobre proporciones construidas de la misma forma que en el conjunto anterior, presentando una mayor aleatoriedad.

3. El tercer conjunto de muestras es generado como el primero mediante muestreo sin reemplazamiento.

4. El cuarto conjunto se genera mediante la permutación aleatoria de los tipos celulares sobre muestras generadas de la misma forma que las del tercer conjunto.

5. El quinto y último es generado mediante muestreo aleatorio de una distribución Dirichlet, lo que da lugar a muestras con distribuciones con mayor mediana en comparación con las anteriores, sobre todo en casos con un número alto de tipos celulares, y gran aleatoriedad.

Respecto a la proporción del total de muestras *bulk* generadas por una u otra distribución, se determinó que una mayoría de muestras aleatorias proporciona mejores resultados. Una posible explicación es que de esta forma la red neuronal conoce un mayor rango de situaciones distintas a pesar de que algunas combinaciones sean raras de encontrar en la realidad, permitiendo identificar mejor a los tipos celulares en función de patrones en las matrices de expresión. Esta información es accesible mediante la función `showProbPlot`, que proporciona diferentes formas de visualización de las proporciones en función del método por el que han sido generadas. En cualquier caso, es posible alterar las proporciones preestablecidas mediante los argumentos `proportions.train` y `proportions.test`. Es posible establecer diferentes proporciones en las distribuciones para los datos de entrenamiento y de test, pero es recomendable utilizar las mismas con el fin de evaluar el modelo de la forma menos sesgada posible. Respecto a la reproducidibilidad, gracias al hecho de que todos los métodos están implementados en R, es posible el establecimiento de una semilla previa a la ejecución de la función.

Finalmente, las células de cada conjunto de datos (entrenamiento y test) son muestreadas aleatoriamente en función del tipo celular al que pertenecen. Cada muestra *bulk* estará compuesta finalmente por un determinado número de células, 100 por defecto, las cuales pertenecerán a un tipo celular u otro en función de las proporciones definidas en la matriz de composición P_{jk}. Toda esta información es almacenada en objetos de la clase *ProbMatrixCellTypes* para su uso por las siguientes funciones.

4. Simulación de perfiles *bulk RNA-seq* y preparación de los datos para entrenamiento

Tras generar las matrices de composición celular P_{kj}, se lleva a cabo la simulación de las muestras *bulk* de entrenamiento y test mediante la función `generateBulkSamples` (Code Box 2.5). Estos perfiles consisten en el sumatorio de los niveles de expresión del gen i en células del tipo celular k de acuerdo a las proporciones establecidas en la matriz P_{kj}. Siguiendo el marco de trabajo expuesto en la Subsección 1.1.4, en esta ocasión, C_{ik} no se corresponde con los niveles de expresión medios de cada tipo celular, ya que las muestras *bulk* se generan mediante el sumatorio de $100 \cdot P_{kj}$ células. Por lo tanto, la Ecuación 1.1 pasa a tener la forma de la Ecuación 2.1, de manera que la matriz T de expresión *bulk* es generada de acuerdo ella:

$$T_{ij} = \sum_{k=1}^{K} \sum_{z=1}^{Z} C_{izk} \tag{2.1}$$

$$\text{de forma que} \begin{cases} i = 1 \dots M \\ j = 1 \dots N \\ Z = 1 \dots 100 \cdot P_{jk} \\ \sum_{k=1}^{K} Z \cdot P_{jk} = 100 \end{cases}$$

donde T_{ij} es el nivel de expresión del gen i en la muestra *bulk* j; C_{izk} es el nivel de expresión del gen i en la célula z del tipo celular k; y P_{kj} es la proporción del tipo celular k en la muestra j. Z representa el número de células que compondrán la proporción del tipo celular k en la muestra *bulk* j. Éstas son muestreadas aleatoriamente en función del tipo celular al que pertenezcan, de forma que cada muestra *bulk* se genera mediante una combinación aleatoria de células de

ese tipo. Nótese que esta ecuación es muy similar a la Ecuación 1.1, exceptuando la ausencia del término de error; el hecho de que $C_{ik} \cdot P_{kj}$ pasa a ser un sumatorio de células de cada tipo celular k de acuerdo a P_{kj}, representando, por tanto, fundamentalmente lo mismo; y que el número total de células que compone cada muestra *bulk* es fijado a 100 por defecto. Este valor es posible modificarlo a través del argumento `n.cells`.

Finalmente, se lleva a cabo la preparación de los datos para el entrenamiento y la evaluación de la red neuronal. Este paso es realizado mediante la función `prepareDataForTraining` (Code Box 2.5). En ella, se ofrece la posibilidad de utilizar para el entrenamiento los perfiles *single-cell*, los perfiles *bulk* o la combinación de ambos mediante el argumento `combine`. En el conjunto de datos de test, en cambio, la función siempre combina ambos tipos de perfiles, pero es posible el filtrado de las células durante la evaluación posterior de los modelos. Después, las muestras son normalizadas y aleatorizadas con el fin de evitar sesgos durante el entrenamiento y las matrices resultantes son transpuestas, ya que la red neuronal requiere que las muestras se localicen en las filas y los genes, en las columnas. Respecto a la normalización, consiste en dos pasos: el cálculo de CPMs (*counts* por millón) y la posterior normalización clásica estableciendo media igual a cero y desviación típica igual a 1. Respecto al primero, se trata de uno de los protocolos de normalización más comunes en experimentos *bulk RNA-seq*. En él, los *counts* son normalizados usando un factor de tamaño proporcional a la profunidad de secuenciación de cada muestra. De esta manera, se evita la existencia de sesgos por muestras *bulk* con mayor cantidad de *counts*. Este protocolo es también el utilizado durante la normalización de nuevas muestras *bulk* para deconvolucionar.

```
## simulation of bulk samples
DDLSChungSmall <- generateBulkSamples(
  DDLSChungSmall,
  threads = 2,
  type.data = "both",
  file.backend = "DDLS_bulk.h5" # if NULL, works in-memory
)
## preparing samples for training (train) and prediction (test)
DDLSChungSmall <- prepareDataForTraining(
  object = DDLSChungSmall,
  type.data = "both",
  combine = "bulk", # or both or single-cell
  file.backend = "DDLS_final.h5" # if NULL, works in-memory
)
```

Code Box 2.5: Ejemplo de la simulación de muestras *bulk* y la preparación de los datos para el entrenamiento y la evaluación. En caso de que `file.backend = NULL`, la función cargará los datos en memoria.

Respecto a la implementación de ambos procesos en el paquete, es importante tener en cuenta que el tamaño que presenta cada matriz simulada puede llegar a ser relativamente grande, ya que suelen tener del orden de 20.000 filas (muestras) y 20.000 columnas (genes). Dado que es necesaria la generación de dos sets de datos, entrenamiento y test, el tamaño de dos matrices de esas dimensiones asciende en conjunto a aproximadamente 6GB como objetos de R ordinarios, lo que puede suponer un problema para ordenadores con *hardware* limitado o para la realización de las operaciones siguientes. Por ejemplo, debido a que el consumo de memoria RAM durante el entrenamiento de la red neuronal es elevado, el mantenimiento de estos objetos en memoria puede hacer imposible hacer los cálculos. Además, en este caso, las matrices no pueden ser representadas como matrices disperas, ya que no presentan una gran cantidad de ceros como sí ocurría en los datos *scRNA-seq*.

Por todo ello, se decidió implementar la opción de utilizar los paquetes DelayedArray

(v0.12.3) (Pagès et al., 2020) y HDF5Array (v1.14.4) (Pagès, 2020) para el almacenamiento de las matrices. Estos paquetes creados por el equipo de Bioconductor proporcionan conjuntamente una API (*Application Programming Interfaces*) de alto nivel para el manejo de ficheros HDF5 desde el entorno de R, de forma que es posible acceder a datos almacenados en disco dinámicamente sin necesidad de mantenerlos cargados en memoria. De este modo, es posible trabajar con datos que no son asumibles por el *hardware* disponible y, además, aprovechar las ventajas que ofrecen los ficheros HDF5, como el almacenamiento de diferentes conjuntos de datos en un solo fichero. El concepto del paquete respecto a las operaciones se basa en tres ideas:

- Operaciones retardadas: las operaciones sobre los datos no son ejecutadas en el momento, sino que las instrucciones son almacenadas en una estructura en forma de árbol en el propio objeto *HDF5Array*. Así, el fichero HDF5 se modificará una sola vez.
- Procesamiento en bloques: permite iterar por bloques sobre los datos de forma que solo son cargados en memoria subconjuntos de los mismos, haciendo que las operaciones se llevan a cabo optimizando al máximo los recursos disponibles.
- Realización: es el último paso en el que las operaciones retardadas son ejecutadas y se produce la escritura de los datos transformados en un fichero HDF5. Gracias a ello, únicamente será necesario la lectura/escritura del fichero una vez. Este proceso también se lleva a cabo mediante procesamiento en bloques.

Por ejemplo, en el caso concreto de dos matrices, una para entrenamiento con 23.555 filas y 18.777 columnas, y otra para test, con mismo número de filas y 12.787 columnas, el tamaño total como objetos ordinarios de R es de 6,5GB (medidas tomadas mediante la función `object.size` del paquete utils). Mediante el uso de esta funcionalidad, el tamaño de los dos objetos pasa a ser de 4,4kB (*kilobytes*), permitiendo trabajar de forma más óptima respecto al uso de memoria RAM en los siguientes pasos y el mantenimiento de toda la información en el objeto *DigitalDLSorter* en la sesión de R interactiva. Los tiempos de ejecución, en cambio, pasan a ser ligeramente mayores debido al acceso a disco de los datos para, por ejemplo, los pasos de realización o el entrenamiento de la red. Sin embargo, es posible aumentar la velocidad de ejecución mediante la modificación de la forma en la que se escriben los datos en el fichero HDF5. Por ejemplo, digitalDLSorteR implementa por defecto que la escritura de las matrices que serán utilizadas para el entrenamiento y la evaluación sea en forma de bloques compuestos por un determinado número de filas, es decir, de muestras (argumento `number.rows`). Dado que, en principio, la red neuronal únicamente accederá a la matriz de expresión génica de esta forma, los tiempos de lectura son muchos menores que los tiempos invertidos utilizando la estructura que HDF5Array ofrece por defecto.

5. Entrenamiento de la Red Neuronal Profunda

Tal y como se ha comentado anteriormente, se decidió utilizar el paquete de R Keras para la implementación de la red. Mediante la función `trainDigitalDLSorterModel`, son llevados a cabo tanto el entrenamiento como la evaluación del modelo. Toda la información producida es almacenada en el *slot trained.model* en forma de un objeto de la clase *DigitalDLSorterDNN* (Code Box 2.6).

La selección de la arquitectura y el establecimiento de los hiperparámetros de la Red Neuronal Profunda fue llevado a cabo por Torroja y Sanchez-Cabo (2019) y son expuestos en el artículo en cuestión. La arquitectura final e implementada en el paquete (Figura B.1) está compuesta por 2 capas ocultas con 200 neuronas cada una. La función de activación seleccionada es

ReLU (*Rectified Linear Unit*), mientras que las neuronas de salida utilizan la función Softmax. El número de neuronas de entrada depende del número de genes considerados en el experimento, mientras que, el de salida, del número de tipos celulares presentes en los datos *scRNA-seq*. El optimizador utilizado es ADAM. Como método de regularización, la red presenta dos capas *dropout* con una tasa de 0.25 para evitar problemas de sobreajuste y dos capas de normalización para reducir posibles sesgos. Finalmente, la función de pérdida utilizada para su entrenamiento es la divergencia de Kullback-Leibler (KLD), aunque es posible su modificación mediante el argumento `loss`. Esta métrica mide la cantidad de información perdida en términos de entropía a la hora de comparar dos distribuciones, por lo que valores más bajos durante el entrenamiento equivaldrán a mejores resultados.

```
## training using train set and prediction over test set
DDLSChungSmall <- trainDigitalDLSorterModel(
  object = DDLSChungSmall,
  batch.size = 128,
  num.epochs = 20,
  val = FALSE # if TRUE, a subset of data is used for evaluation during training
)
```

Code Box 2.6: Ejemplo del entrenamiento de la red neuronal con los datos de entrenamiento y la predicción sobre los datos de test.

Respecto a la implementación de la función `trainDigitalDLSorterModel`, debido a la potencial cantidad de datos que pueden generarse en los pasos anteriores, se decidió implementar los procesos de entrenamiento y predicción mediante el uso de generadores de datos. Estas funciones se encargan de devolver un determinado número de muestras tras cada llamada de forma ilimitada, de manera que puedan llevarse a cabo tantas llamadas a la función como sean necesarias. En este caso, se han implementado generadores en R junto con el uso de las funciones `fit_generator` y `evaluate_generator` de Keras. El objetivo consiste en no cargar todas las muestras en memoria durante el entrenamiento o la evaluación, sino únicamente aquellas que vayan a ser utilizadas en cada época. Mediante el argumento `batch.size`, es posible determinar el número de muestras utilizadas para el entrenamiento en cada época. Esta implementación, junto con el uso de los ficheros HDF5 como *back-end* de las muestras, hacen de este paso una tarea relativamente ligera a nivel computacional y con tiempos de ejecución menores que en el caso de haber sido implementada a través de la lectura de ficheros tabulados.

6. Cálculo de métricas de evaluación del modelo

Tras el entrenamiento del modelo y la predicción llevada a cabo sobre los datos de test, un paso imprescindible es la evaluación del modelo mediante métricas de error. Sin embargo, a pesar de estar utilizando una arquitectura típica para un problema de clasificación por el uso de la función Softmax en la capa de salida, la evaluación no debe estar basada en métricas como la precisión. Ésta únicamente informa del error cometido respecto a la clase mayoritaria en cada muestra, tratándose de una visión incompleta considerando el problema que se quiere resolver. De hecho, es posible que, durante el entrenamiento, esta métrica no alcance valores extremadamente altos debido a las proporciones mayoritariamente aleatorias de las muestras *bulk* generadas. Es por ello que, para conocer la precisión del modelo respecto al problema de deconvolución y poder determinar si existe alguna tendencia a cometer errores mayores en ciertas condiciones, es necesario el cálculo de métricas que tengan en cuenta todas las proporciones.

Para ello, digitalDLSorteR proporciona la función `calculateEvalMetrics`. Por defecto, lleva a cabo el cálculo de las dos métricas de error clásicamente utilizadas para la evaluación de las herramientas de deconvolución: el error absoluto y el error cuadrático (Mohammadi et al.,

2017) junto a sus versiones porcentuales. Estas metricas son calculadas para cada una de las muestras que componen el conjunto de datos de entrenamiento mediante la comparación de las proporciones celulares reales y las predichas. Además, la función lleva a cabo por defecto el cálculo de los valores medios correspondientes a la agregación de las muestras en función de varios criterios: el tipo celular (`CellType`), el número de tipos celulares distintos en las muestras (`nMix`) y el rango de probabilidad en el que se encuentre un tipo celular (`pBin`). Toda esta información puede representarse gráficamente mediante la función `distErrorPlot`, con la que es posible observar cómo se distribuyen los errores en función de estas variables. Cabe destacar la posibilidad de fragmentar los gráficos en función del tipo celular o del número de tipos celulares presentes en las muestras, permitiendo así una mejor visualización de los resultados. Además, en cada gráfico es mostrado el error medio en función de la variable que se haya seleccionado en forma de anotación (Figura 4.5). También es posible representar los errores medios en función de estas variables junto con su dispersión mediante la función `barErrorPlot`.

```
## calculation of absolute error and squared error
DDLSChungSmall <- calculateEvalMetrics(DDLSChungSmall)
## distribution of absolute errors by cell type
distErrorPlot(
  DDLSChungSmall,
  error = "AbsErr", # or ppAbsErr, RsqrErr...
  facet.by = "CellType", # or nMix or NULL...
  color.by = "nMix", error.labels = TRUE
)
## correlation between expected and prediction
corrExpPredPlot(
  DDLSChungSmall, corr = "both", ## pearson or CCC
  facet.by = "CellType", color.by = "CellType",
  filter.sc = FALSE
)
## Bland Altman agreement plot
blandAltmanLehPlot(
  DDLSChungSmall, facet.by = "CellType",
  color.by = "CellType",
  log.2 = TRUE, density = TRUE
)
```

Code Box 2.7: Ejemplos de funciones relacionadas con el cálculo y la representación gráfica del desempeño del modelo sobre los datos de test.

Otra de las métricas típicamente utilizadas y disponible en el paquete es el coeficiente de correlación de Pearson (R), con el que es posible medir la dependencia linear entre las proporciones reales y las proporciones predichas. Además, se ha incluido también la posibilidad calcular el coeficiente de correlación de concordancia (CCC) (Lin, 1989) (Figura 4.6.c). Esta medida es sensible no solo a desviaciones de la linearidad, sino a desviaciones de la diagonal resultante en el caso de que los procentajes reales y predichos fueran iguales (identidad). Por lo tanto, en el caso de que los resultados presenten una gran correlación linear, pero se esté produciendo una sobreestimación o una subestimación, CCC bajará. Viene dada por la Ecuación 2.2, en la que $\overline{x}$ es la media, s_x^2 la varianza y s_{xy} la covarianza. Ambas métricas están disponibles mediante la función `corrExpPredPlot`, siendo presentadas junto al correspondiente gráfico de dispersión en forma de anotación.

$$CCC = \frac{2s_{xy}}{s_x^2 + s_y^2 + (\overline{x} - \overline{y})^2} \tag{2.2}$$

Finalmente, una última alternativa es el uso de la función `blandAltmanLehPlot` (Figura 4.7.c). Esta función construye el gráfico de Bland-Altman (Altman y Bland, 1983), el cual permite medir el grado de acuerdo entre dos variables mediante la representación de la media en el eje X y la diferencia entre ambas variables en el eje Y. De esta forma, se lleva a cabo la cuantificación de la diferencia media entre las dos variables y, además, se establecen dos umbrales de concordancia dados por 1,96 veces la desviación típica de las diferencias, entre los que se espera que se incluyan el 95 % de las diferencias. También es posible su representación en escala logarítmica.

Todas estas funciones han sido implementadas utilizando el paquete de representación gráfica ggplot2 (Wickham, 2016b). En la Code Box 2.7, es posible encontrar algunos ejemplos sobre su uso, así como los gráficos resultantes en la viñeta del paquete. Todos los gráficos presentados a lo largo del Capítulo 4 han sido construidos mediante estas funciones, mostrando así no solo los resultados, sino el tipo de gráficos que generan. Tras la exposición de cada función, se ha referenciado una figura presente en el documento a modo de ejemplo.

7. Deconvolución de nuevas muestras *bulk RNA-seq*

El último paso consiste en la carga de nuevas muestras *bulk RNA-seq* en el objeto *DigitalDLSorter* para su deconvolución. Para ello, se ha implementado la posibilidad de cargar nuevos datos *bulk* desde ficheros en disco con la función `loadDeconvDataFromFile` o desde objetos *SummarizedExperiment* con la función `loadDeconvDataFromSummarizedExperiment`. En ambos casos, los datos son almacenados en el *slot deconv.data*. Después, es posible utilizar la función `deconvDigitalDLSorterObj` para la deconvolución de los datos utilizando el modelo contenido en el objeto. Los resultados son almacenados en el *slot deconv.results* (Code Box 2.8). Al igual que en la función `deconvDigitalDLSorter`, el método de normalización empleado consiste en una primera transformación a CPMs y la posterior normalización estableciendo media igual a cero y desviación típica igual a 1. Además, también es posible el uso de los argumentos `simplify.set` y `simplify.majority`, de forma que los resultados se almacenan junto a las predicciones originales como una lista.

```
## new SummarizedExperiment object
TCGA.breast <- SummarizedExperiment(
  assay = list(counts = TCGA.breast.small)
)
## load SE object into DigitalDLSorter object
DDLSChungSmall <- loadDeconvDataFromSummarizedExperiment(
  object = DDLSChungSmall,
  se.object = TCGA.breast,
  name.data = "TCGA.breast"
)
## deconvolution using this data
DDLSChungSmall <- deconvDigitalDLSorterObj(
  object = DDLSChungSmall,
  name.data = "TCGA.breast",
  normalize = TRUE
)
## see results
barPlotCellTypes(DDLSChungSmall, name.data = "TCGA.breast")
```

Code Box 2.8: Ejemplo de la deconvolución de nuevas muestras *bulk RNA-seq* desde el objeto *DigitalDLSorter*.

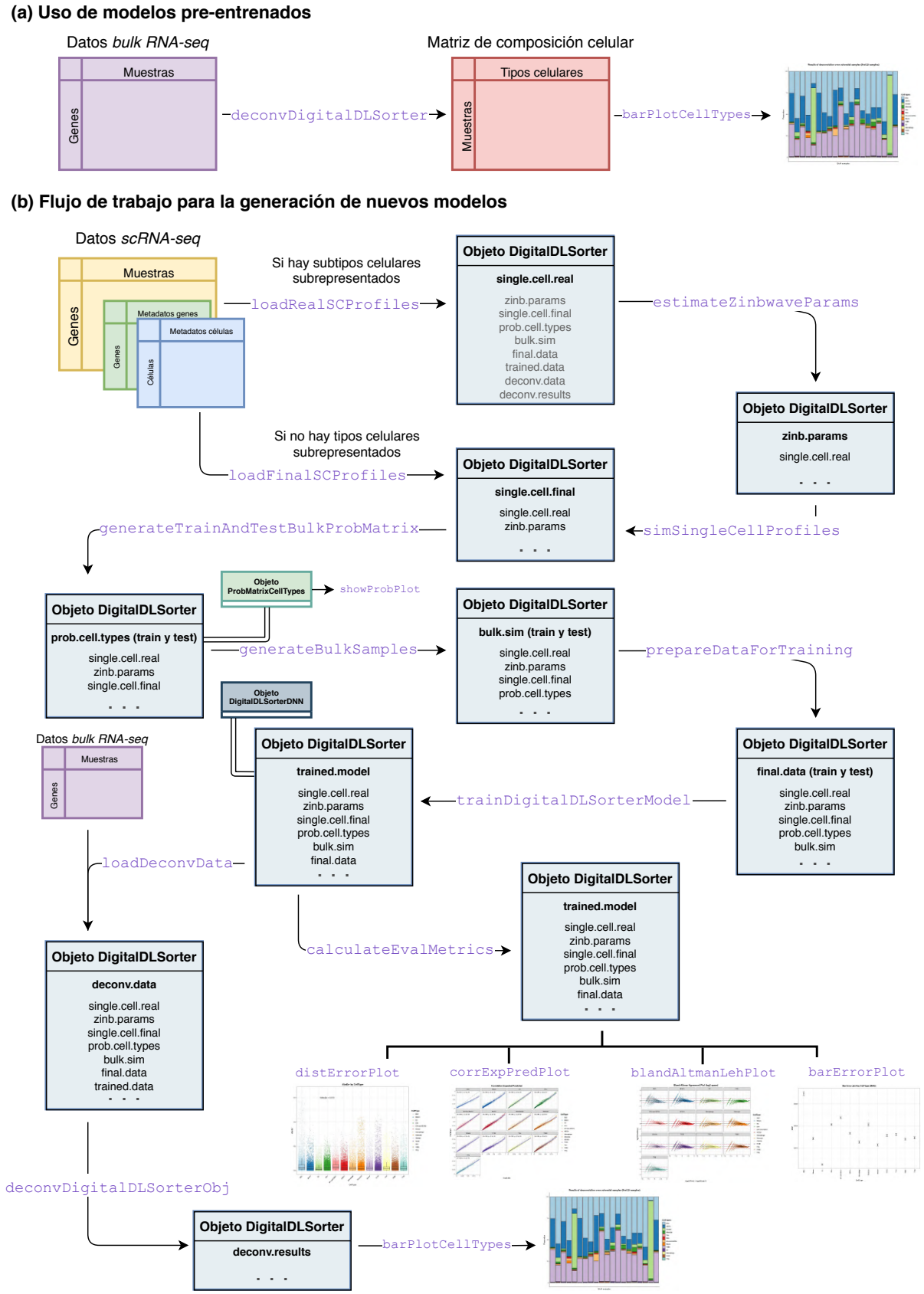

Figura 2.2: Diagrama resumen del flujo de trabajo del paquete digitalDLSorteR. **(a):** Uso de modelos preentrenados ofrecidos por el paquete. **(b):** Flujo de trabajo de las principales funcionalidades para la construcción de nuevos modelos a partir de datos *scRNA-seq*.

3

Análisis de datos *scRNA-seq* de muestras de cáncer de mama

3.1. Introducción

Como se ha expuesto a lo largo de la Sección 1.1, la naturaleza heterogénea de los tumores y el papel clave de las células inmunes en diferentes aspectos del cáncer hacen necesaria su exploración a través de toma de datos *scRNA-seq* o mediante herramientas de deconvolución. Prestando especial atención al cáncer de mama, en el presente Trabajo Fin de Máster se ha llevado a cabo el análisis de datos *scRNA-seq* de pacientes con la enfermedad procedentes del artículo Chung et al. (2017). El objetivo es la caracterización de las células presentes en el experimento para construir un modelo de deconvolución mediante el paquete digitalDLSorteR.

3.2. Datos

Los datos utilizados proceden del artículo Chung et al. (2017), en el que los autores llevaron a cabo el análisis de 11 pacientes con cáncer de mama de los que capturaron el perfil transcriptómico de 549 células. Los pacientes representan los 4 subtipos intrínsecos de cáncer de mama expuestos en la Subsección 1.1.2 y fueron caracterizados mediante el uso de marcadores. En la Tabla 3.1, se muestra la información relativa a cada uno de ellos: el tipo de cáncer de mama que presentan, la procedencia de los datos y el número de células correspondientes a cada uno. El protocolo de aislamiento de las células y de amplificación y secuenciación del material genético es descrito en el artículo.

Respecto a la disponibilidad y obtención de los datos, los autores publicaron la matriz de expresión génica procesada correspondiente a cada una de las células en la base de datos *Gene Expression Omnibus* (GEO) con el identificador GSE75688 y, además, las lecturas en la base de datos SRA con el identificador SRP066982. Dado que se disponía de las lecturas iniciales de los tránscritos, se decidió utilizar estos datos con el objetivo de llevar a cabo el análisis desde el principio y evitar el uso de las matrices de expresión procesadas por los autores.

ID Paciente	Subtipo	Tejido	# total de células
BC01	Luminal A	Tumor primario	26
BC02	Luminal A	Tumor primario	56
BC03	Luminal B	Tumor primario	37
BC04	HER2	Tumor primario	59
BC05	HER2	Tumor primario	77
BC06	HER2	Tumor primario	25
BC07	TNBC	Tumor primario	51
BC08	TNBC	Tumor primario	23
BC09	TNBC	Tumor primario	60
BC10	TNBC	Tumor primario	16
BC11	TNBC	Tumor primario	11
BC03LN	Luminal B	Tejido linfático	55
BC07LN	TNBC	Tejido linfático	53

Tabla 3.1: Información de las muestras. Nota: las muestras BC03LN y BC07LN corresponden a células de tejido linfático metastásico de los pacientes BC03 y BC07, respectivamente.

3.3. Alineamiento y cuantificación

3.3.1. Obtención y filtrado de secuencias fastq

Tras la descarga de las lecturas en formato sra, se obtuvieron los ficheros fastq correspondientes mediante el comando `fastq-dump` del set de herramientas sratools. Después, se llevó a cabo su procesamiento con el fin de mantener únicamente aquellas con suficiente buena calidad. Para ello, se utilizó el *software* cutadapt (Martin, 2011), con el que se realizaron varios pasos de procesamiento:

- Eliminación de adaptadores de la preparación de las librerías con los argumentos `-m 30` (longitud mínima) y `-O 3` (solapamiento mínimo). Las secuencias utilizadas como objetivo para su eliminación fueron los adaptadores del kit de preparación de librerías Nextera XT DNA Sample Prep (FC-131-1024, Illumina) utilizado por Chung et al. (2017).
- Eliminación de colas Poli(A) mediante el argumento `-a` para bases repetidas. Se eliminaron cadenas con repeticiones de 25, 50 y 70 adeninas.

En cada uno de los pasos, se llevó a cabo la medición de las métricas de calidad de las lecturas mediante el *software* fastqc (Andrews et al., 2012) para mantener un control durante el proceso.

3.3.2. Construcción de la referencia para el alineamiento

Para la cuantificación de la expresión génica, primeramente se llevó a cabo la construcción del genoma de referencia contra el que fueron alineadas las lecturas mediante el *software* RSEM (Li y Dewey, 2011) con el comando `rsem-prepare-reference`. Se utilizó la versión GRCh38.p13 del genoma humano junto con la versión 33 de las anotaciones GENCODE, la más reciente en el momento de construcción.

En los experimentos *RNA-seq*, el objetivo es la cuantificación de la expresión génica, por lo que las lecturas de partida consisten en ADNc procedente de secuencias de ARN enriquecidas con cola de Poli(A), es decir, ARNm. Por ello, se decidió filtrar del fichero gtf aquellas entradas

que no correspondieran a exones. Además, también se eliminaron las entradas de pseudo-genes y tránscritos de pequeño tamaño como micro-ARNs, ARNs pequeños nucleares, etc. El objetivo fue evitar ruido durante el proceso de alineamiento mediante la reducción de la variabilidad de las secuencias de referencia, manteniendo únicamente aquellas interesantes para el experimento. Además, se llevó a cabo la adición de las secuencias *spike-ins* (ThermoFisher) tanto a las anotaciones como al fichero fasta del genoma de referencia. El proceso se implementó mediante el uso de comandos awk.

3.3.3. Alineamiento y cuantificación de los tránscritos

Una vez obtenidas las lecturas definitivas tras los pasos de procesamiento y construida la referencia, se alinearon las lecturas y cuantificaron los tránscritos con el fin de obtener los niveles de expresión de cada gen en las células. Todo ello se realizó con el *software* RSEM mediante el comando `rsem-calculate-expression`. RSEM permite llevar a cabo tanto la cuantificación como el alineamiento de manera conjunta, de forma que, para el alineamiento, se decidió utilizar STAR (Dobin y Gingeras, 2015) mediante el uso del parámetro `-star`. Los resultados se almacenaron en formato bam.

Respecto a los parámetros utilizados para los pasos relacionados con la cuantificación, se estableció que la longitud de los fragmentos siguiera una distribución normal con media igual a 180 pares de bases y desviación estándar igual a 50. Además, se utilizó el argumento `-estimate-rspd` con el que RSEM lleva a cabo la estimación de la distribución del inicio de las lecturas. El resto de parámetros utilizados fueron los que presenta por defecto. El fundamento de la herramienta se basa en un algoritmo de esperanza-maximización para discernir con qué probabilidad un transcrito ha podido producir esa lectura alineada, dando lugar, por tanto, a una matriz de *pseudo-counts*. Los *counts* (niveles de expresión procesados) son devueltos tanto en FPKMs (Fragmentos por millón de kilobases) como en TPMs, habiéndose seleccionado los últimos por ser una unidad de cuantificación más consistente entre muestras (Wagner et al., 2012), lo que tiene especial relevancia en los experimentos *scRNA-seq*.

3.4. Análisis de los datos

Tras la obtención de los niveles de expresión para cada célula, se llevó a cabo el procesado de los ficheros devueltos por RSEM y la construcción de la matriz de expresión génica mediante R. A partir de este punto, el análisis se puede dividir en dos pasos principales: el preprocesamiento de los datos y los análisis posteriores.

3.4.1. Preprocesamiento: filtrado y normalización

Antes de comenzar con el análisis, es necesario llevar a cabo el cálculo de métricas de calidad con el fin de asegurar que los datos de cada célula corresponden realmente a células únicas viables. Es común encontrar perfiles de expresión que pertenecen a células de mala calidad, células con membranas rotas o dobletes (agrupaciones de más de una célula). Las variables utilizadas habitualmente para la medición de la calidad son el número de *counts* por célula, el número de genes detectados por célula, la fracción de *counts* perteneciente a genes mitocondriales, la fracción de *counts* perteneciente a genes ribosómicos y, en el caso de que el experimento los presente, la fracción de *counts* pertenecientes a *spike-ins* (lecturas de calibración de la secuenciación). La distribución de estas variables puede ser analizada con el fin de eliminar células atípicas mediante umbrales.

En el caso de los datos de estudio, de las 549 células de partida, fueron filtradas 61. Debido que la suma de los TPMs totales en cada célula es la misma, se utilizó el número de lecturas por célula para filtrar por profundidad de secuenciación. En la Figura 3.1, se pueden observar las distribuciones de cada una de estas variables agrupadas por muestra y los umbrales utilizados. En la Figura 3.2, se ha representado el gráfico de dispersión resultante de enfrentar el número de lecturas contra el número de genes detectados, pudiéndose observar cómo las células filtradas se corresponden con puntos relativamente atípicos.

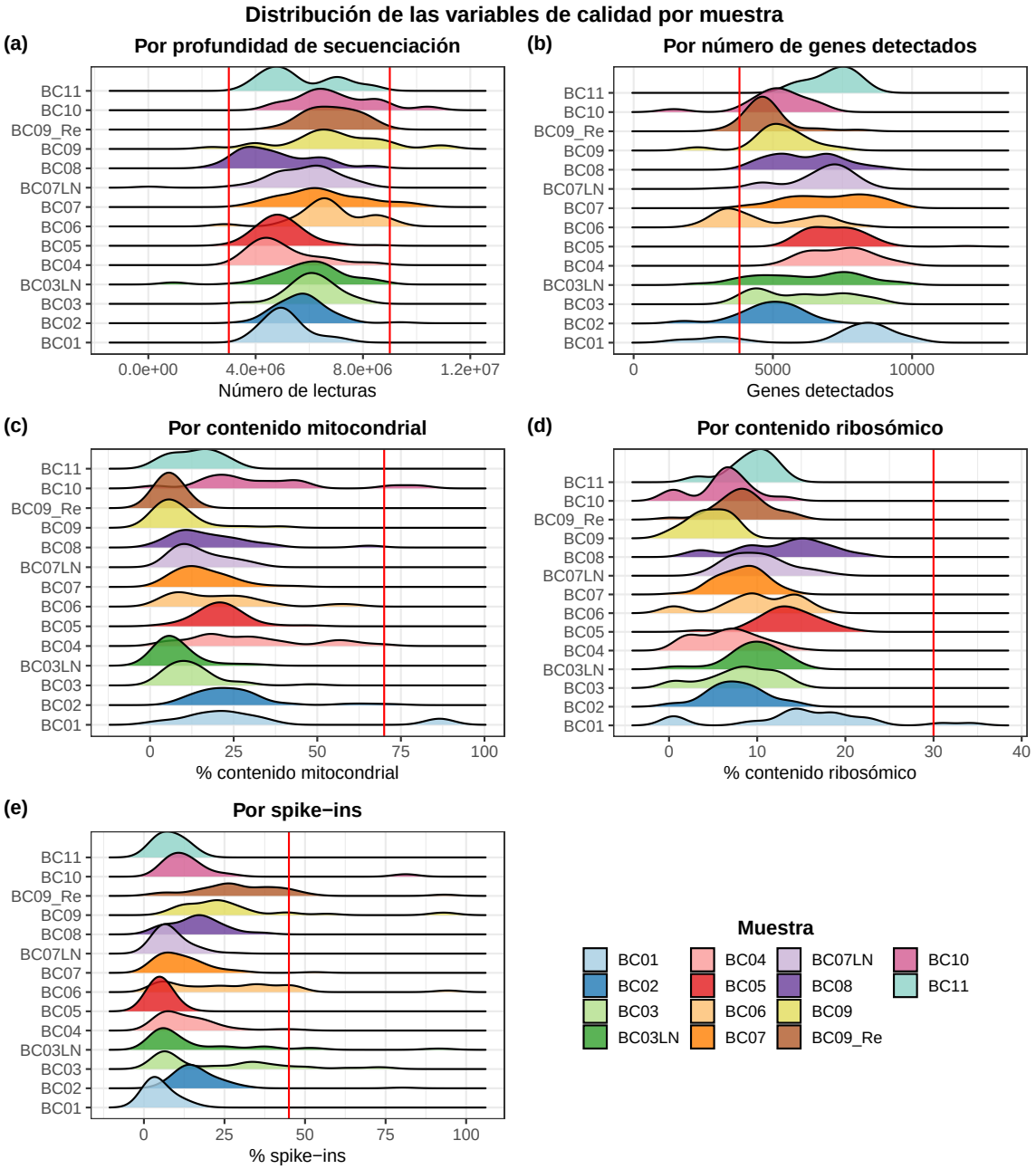

Figura 3.1: Distribución de las variables de calidad en las células agrupadas por muestra. Los umbrales utilizados en cada caso son: **(a):** Filtrado por profundidad de secuenciación (número de lecturas): $3 \cdot 10^6$ - $9 \cdot 10^6$. **(b):** Filtrado por número mínimo de genes detectados: 3.800. **(c):** Filtrado por fracción máxima de contenido mitocondrial: 70 %. **(d):** Filtrado por fracción máxima de contenido ribosómico: 30 %. **(e):** Filtrado por fracción máxima de contenido en s*pike-ins*: 45 %.

Células con baja profundidad de secuenciación, un número bajo de genes detectados y fracciones elevadas de contenido mitocondrial, es probable que se correspondan con células cuya

membrana plasmática se ha roto, habiéndose degradado el ARNm y detectándose únicamente los genes mitocondriales que han permanecido intactos. Por el contrario, células con un gran número de genes detectados o gran cantidad de lecturas, es probable que se deban a dobletes. Respecto al contenido en *spike-ins*, células con elevadas fracciones presentarán una heterogeneidad relativamente baja debido a que el ARNm de partida capturado será bajo. Esto puede ser indicativo de estrés celular que puede resultar en la degradación del ARN, por lo que es recomendable su filtrado.

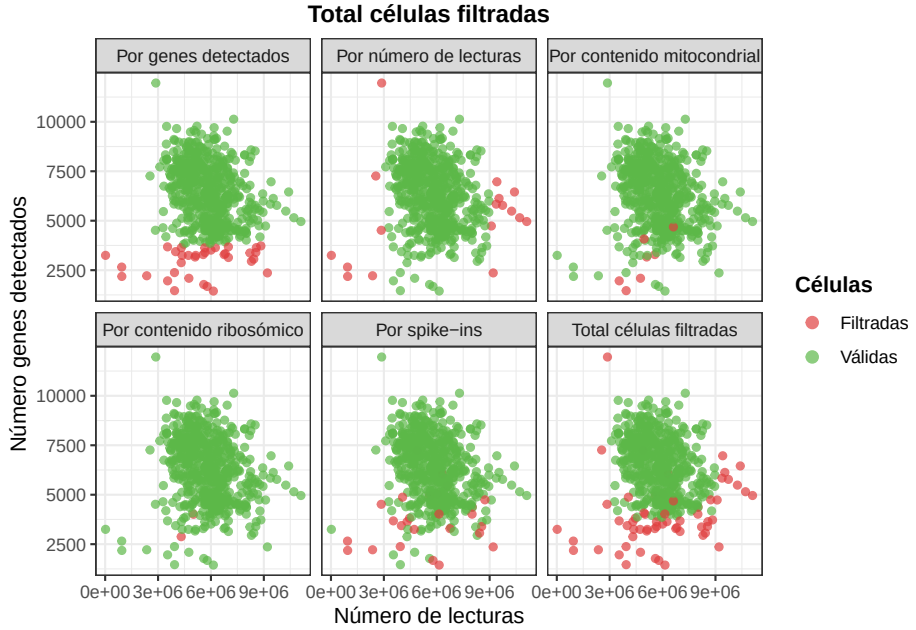

Figura 3.2: Gráfico de dispersión de número de lecturas contra número de genes detectados. Las células filtradas (en rojo) son mostradas en función de la variable por la que han sido eliminadas.

En la Tabla 3.2, se muestra el número de células filtradas por cada uno de los métodos, mientras que, en la Figura B.2 del Material suplementario B, el diagrama de Venn correspondiente donde se puede observar el número de células filtradas por varios criterios a la vez.

	Células filtradas	**Células válidas**
Por genes detectados	35	514
Por número de lecturas	17	532
Por contenido mitocondrial	7	542
Por contenido ribosómico	2	547
Por spike-ins	26	523
Filtrado final	61	488

Tabla 3.2: Número de células filtradas mediante las variables de calidad utilizadas.

Tras la obtención de las 488 células definitivas, el siguiente paso consistió en la normalización de los datos. Como se ha comentado en la Subsección 1.1.3, los datos *scRNA-seq* presentan una gran cantidad de ruido técnico. Por ello, para hacer comparables las células y eliminar la heterogeneidad técnica mientras se preservan las diferencias biológicas, es imprescindible la normalización de los niveles de expresión. Dentro del amplio espectro de métodos de normalización para *scRNA-seq* que existen, en el presente estudio se ha aplicado el método propuesto por los autores del paquete Seurat (Stuart et al., 2019) llamado sctransform (Hafemeister y Satija, 2019). Utiliza un modelo de regresión binomial negativa regularizado para ajustar los

datos y eliminar la variabilidad técnica y los centra para que presenten media cero entre células. Dado que el método parte de la premisa de que los datos consisten en UMIs (*Unique molecular identifier*), se llevó a cabo la transformación de los TPMs a esta unidad mediante el algoritmo Census implementado en el paquete monocle (Qiu et al., 2017). La diferencia entre ambas unidades radica en que los TPMs son una medida relativa de la expresión, mientras que, por el contrario, los UMIs son una medida absoluta a nivel de tránscrito, no a nivel de *counts*. Esto la convierte en una medida más precisa con respecto a la representación de los niveles de expresión.

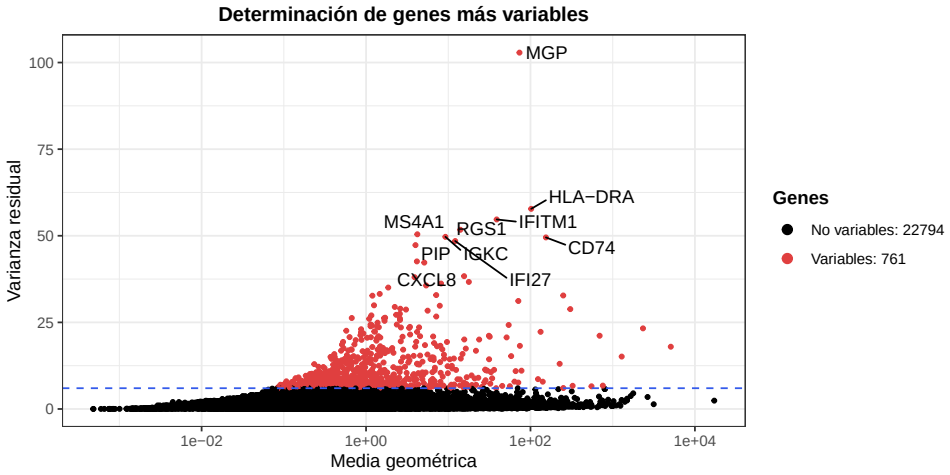

Figura 3.3: Gráfico de dispersión de la media geométrica contra la varianza residual de los niveles de expresión génica normalizados. Umbral de varianza residual utilizado: 6 (línea discontinua azul).

El número de genes más variables detectados estableciendo un umbral de varianza residual igual a 6 fue 761 de los 22.794 de partida (Figura 3.3), siendo los utilizados en los análisis posteriores. Se probaron otros umbrales con el fin de comparar los clústeres obtenidos en los siguientes pasos, siendo éste el que mejor desempeño ofreció. Se puede observar que en el top 10 genes más variables se encuentran genes relacionados con el sistema inmune, como CXCL8 o HLA-DRA, dando a entender que el número de células inmunes presentes en el conjunto de datos es elevado.

3.4.2. Análisis posteriores: reducción de la dimensionalidad y clusterización

Una vez se establecieron las células válidas y los genes más variables, el siguiente objetivo fue la clusterización de las células para su posterior caracterización. Para ello, se utilizó la metodología que ofrece Seurat basada en un primer paso de reducción de dimensionalidad y, en base a las nuevas variables obtenidas, la clusterización. A continuación, se hará un breve resumen del método y se mostrarán los resultados obtenidos tras su aplicación en el presente conjunto de datos.

El primer paso consiste en reducir la dimensionalidad original, en este caso los 761 genes más variables, con el objetivo de disminuir el extenso ruido técnico presente en las características de los datos *scRNA-seq* mientras se preserva la señal de interés. Como se ha comentado anteriormente, las células, a pesar de que puedan pertenecer a una misma población, presentan inconsistencias respecto a los niveles de expresión de sus genes. Por ello, en lugar de utilizar los genes como variables individuales, se lleva a cabo el cálculo del Análisis de Componentes Principales (PCA) para el uso de los componentes principales como 'metavariables'. De esta forma, cada componente principal (PC) combina la información de un conjunto de genes correlacionados linearmente entre sí, por lo que se espera que los PCs sean más consistentes entre células.

Aquellos que expliquen mayor cantidad de varianza representarán una compresión robusta de los datos originales y serán los utilizados en el paso de clusterización.

Respecto a la determinación del número de PCs utilizados, en el presente análisis se seleccionaron los primeros 12 PCs. La decisión se basó en uso del gráfico 'de codo' (Figura 3.4.a), así como en el método JackStraw (Figura 3.4.b) que ofrece el paquete Seurat. Éste último consiste en el remuestreo aleatorio de un subconjunto de los datos y la reejecución de la PCA, construyendo una distribución nula a partir de las puntuaciones de las características con respecto a los componentes principales. El proceso es repetido 100 veces con el fin de determinar la significancia de cada PC, que será la media de los 100 p-valores empíricos.

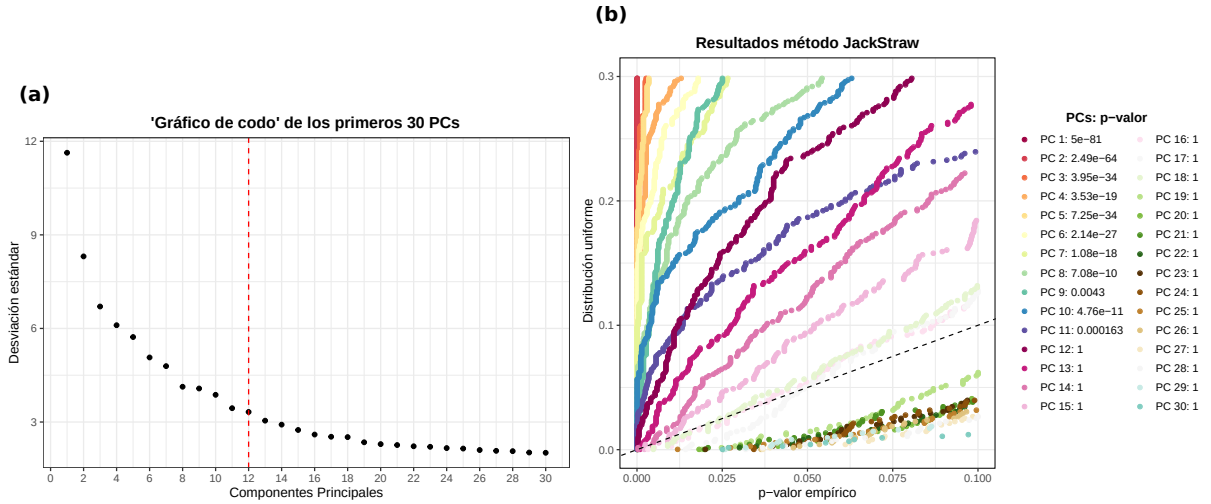

Figura 3.4: (a): Gráfico 'de codo' correspondiente a los primeros 30 PCs (línea roja discontinua indica el número de PCs considerados en la clusterización). **(b):** Gráfico JackStraw: comparativa de los p-valores empíricos con respecto a una distribución uniforme (línea discontinua). Primeros 11 PCs significativos.

Con las células representadas en un espacio N-dimensional más bajo, en este caso 12 PCs, el siguiente paso consiste en su clusterización. Para ello, Seurat implementa un método basado en grafos muy similar al creado por Levine et al. (2015) para datos de citometría de masas. El algoritmo consiste en el siguiente procedimiento:

1. En primer lugar, lleva a cabo la construcción de un grafo ponderado mediante la aplicación del algoritmo k-vecinos más cercanos (kNN) en base a las distancias euclídeas en el espacio N-dimensional definido por la PCA.

2. Después, con el fin de corregir el hecho de que todas las células presenten el mismo número de vecinos, redefine los pesos del grafo mediante el cálculo del índice de Jaccard para cada par de células en función del número de vecinos que comparten.

3. Finalmente, para clusterizar las células, se lleva a cabo la búsqueda de comunidades en el grafo. Una comunidad consiste en una agrupación de nodos que presentan una mayor densidad de conexión entre ellos, la cual, en este caso, está definido por los pesos. Para ello, Seurat ofrece diversos métodos, habiéndose seleccionado el método de Louvain con refinamiento multinivel.

Este proceso requiere del ajuste de dos hiperparámetros: el número de k vecinos para kNN, el cual no produce grandes cambios en los resultados finales gracias a la corrección mediante el índice de Jaccard, y un parámetro de resolución requerido para la búsqueda de las comunidades. En este caso, se utilizó $k = 5$ y se probaron diferentes valores de resolución, estableciéndose finalmente en 0.5, con el que se encontraron 15 clústeres. Además, con el fin de obtener una

visualización más fiel a la distribución real de los datos en el espacio 12-dimensional definido por la PCA, se llevó a cabo la aplicación del algoritmo de reducción de dimensionalidad no linear tSNE (*t-Distributed Stochastic Neighbor Embedding*) con perplejidad igual a 150. Como se puede observar en la Figura 3.5, los clústeres encontrados co-localizan con la visualización salvo en algunos casos (clúster C9). Cada uno de ellos está compuesto por células similares entre sí, por lo que pueden ser interpretados como poblaciones del mismo tipo celular.

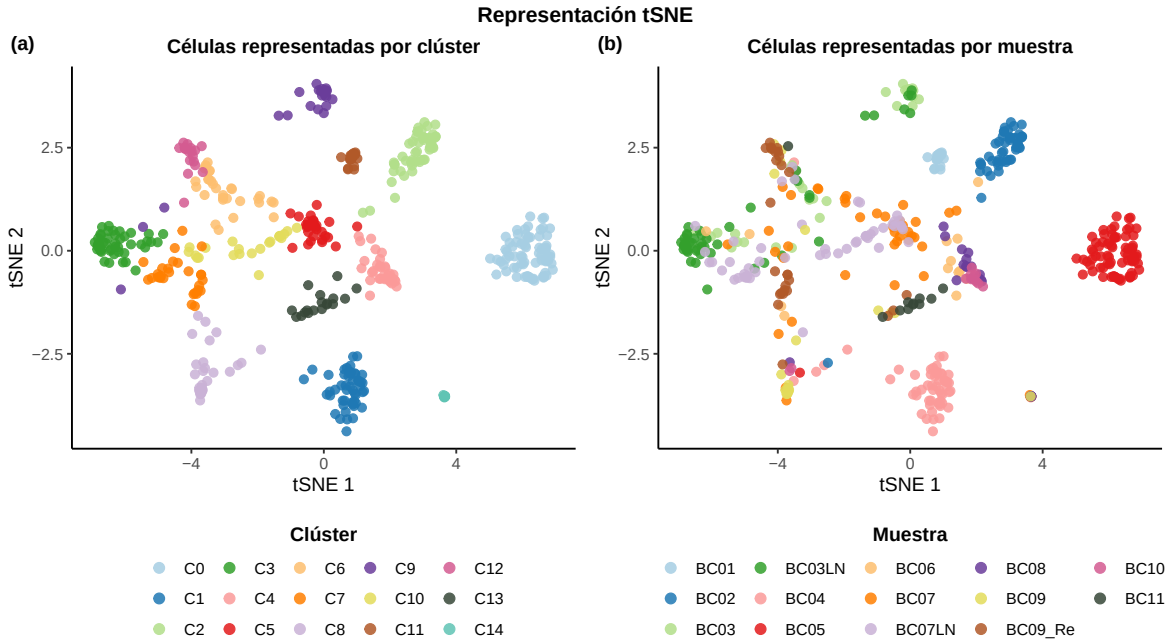

Figura 3.5: Representación tSNE de las células del experimento completo. **(a):** En función de los clústeres encontrados. **(b):** En función de las muestras de las que proceden.

Cabe destacar que, a priori, parece que existe un sesgo respecto a las células procedentes de las muestras BC02, BC04 y BC05 por aparecer en su mayoría agrupadas en clústeres independientes (Figura 3.5). Sin embargo, durante la identificación de las células tumorales/no tumorales, se demuestra que estas diferencias no se deben a ruido técnico, sino a que estas muestras están compuestas por una mayoría de células neoplásicas (Figura 3.7). Respecto a la razón por la que aparecen en clústeres diferentes, se debe a que la firma transcriptómica de los subtipos moleculares del cáncer de mama no es la misma, pudiendo haber, incluso, una alta heterogeneidad dentro de cada subgrupo, tal y como se comenta en la Subsección 1.1.2.

3.5. Caracterización de las células

Las células de este experimento pueden dividirse principalmente en dos conjuntos: células tumorales, que pueden pertenecer a los diferentes subtipos intrínsecos de cáncer de mama; y células no tumorales, entre las que se encuentran tanto células del sistema inmune como células estromales (fibroblastos, células endoteliales, células epiteliales...). Por ello, el primer paso para la caracterización consistió en la separación de estos dos grupos. Después, se llevó a cabo el análisis y la identificación de las células no tumorales con el mayor nivel de resolución posible.

3.5.1. Separación de células tumorales y células no tumorales

La aproximación utilizada para la separación entre ambos grupos fue la presentada por los autores del artículo del que proceden los datos (Chung et al., 2017). Se basa en el uso de los

patrones de expresión génica cromosómicos que caracterizan a las células tumorales. Debido a la inestabilidad génica de estas células, presentan mutaciones CNVs (*Copy Number Variations*) que pueden ser recapitulados a través de los niveles de expresión. Mediante la comparación de las células con los niveles de expresión de tejido mamario normal, es posible la detección de aquellas con patrones anormales.

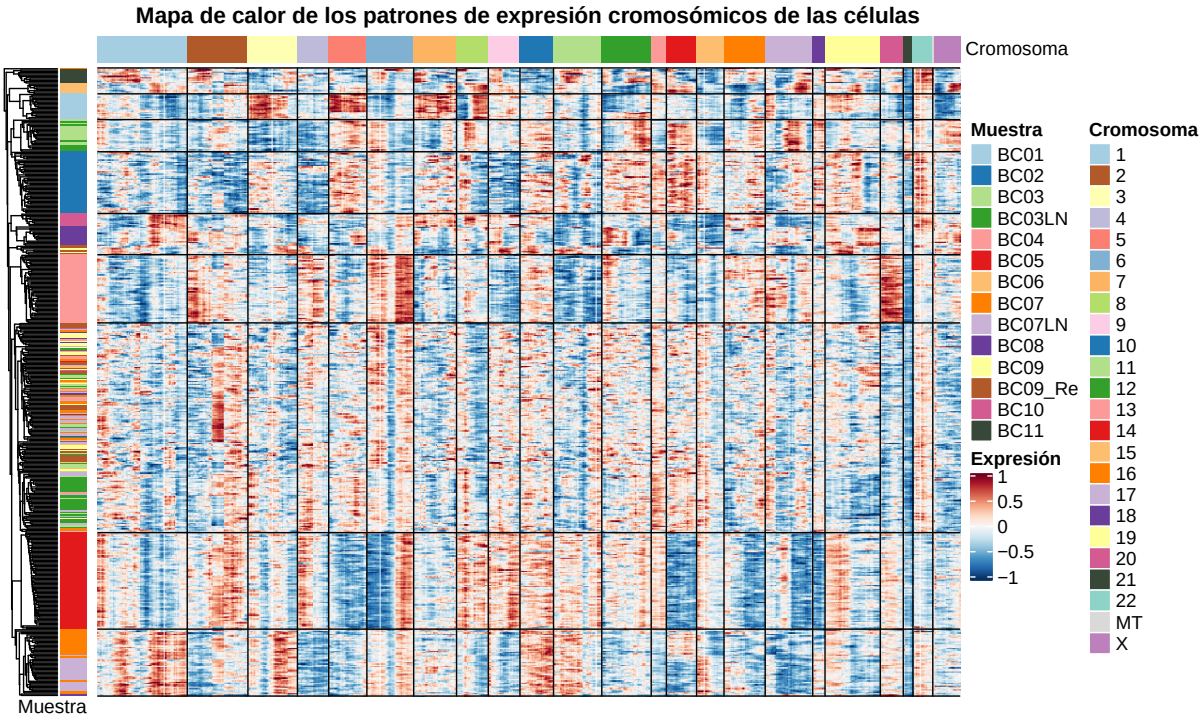

Figura 3.6: Mapa de calor de los patrones de expresión cromosómicos de las células del experimento. Las líneas verticales separan los patrones de expresión de cada cromosoma. Las líneas horizontales separan agrupaciones principales de células tumorales con el fin de mejorar la visualización.

Para ello, en primer lugar, se descargaron los perfiles de tejido mamario normal del portal GTEx (*Genotype-Tissue Expression*) (Carithers et al., 2015) que utilizaron los autores. Los datos fueron transformados a escala logarítmica ($log_2(TPM+1)$) y se calcularon los niveles de expresión medios y la desviación estándar de los genes entre muestras. Después, se calcularon los Z-scores para cada gen mediante la normalización de los perfiles *single-cell* con los niveles de expresión medios de las muestras *bulk*. Los genes fueron ordenados en función de su localización cromosómica y su posición de inicio y los patrones de expresión cromosómicos fueron estimados a partir de los niveles medios en ventanas de 150 genes en comparación con los valores medios ajustados. Finalmente, las células fueron clusterizadas mediante clústering jerárquico utilizando el método UPGMA. En la Figura 3.6, se muestran las diferencias detectadas en los patrones de expresión de las células a lo largo de los cromosomas. Tras la clusterización, se pueden observar grandes agrupaciones de células tumorales procedentes de las mismas muestras presentando patrones de expresión muy característicos con algunas zonas delecionadas (con niveles de expresión más bajos) y otras duplicadas (con niveles de expresión más altos). En la zona media del mapa de calor, se observa una agrupación de células procedentes de diferentes muestras sin patrones CNV aparentes, tratándose de las células no tumorales. De esta forma, se determinaron como tumorales las células de los clústeres con patrones de expresión anormales, siendo 325 de las 488 originales. Dado que esta fue la aproximación utilizada por Chung et al. (2017), se compararon los resultados con los obtenidos por los autores, obteniendo un grado de acuerdo del 95 %.

En la Figura 3.7.a, se muestra la clasificación tumoral/no tumoral de las células representadas mediante tSNE. Salvo algunos casos en los que células tumorales aparecen en los mismos

clústeres que las no tumorales, en general, tanto la localización en la representación mediante tSNE como la clusterización (Figura 3.5.a) concuerdan con las diferencias detectadas utilizando los patrones de expresión cromosómicos. Ambos análisis son independientes entre sí, por lo que el gran grado de acuerdo que presentan confirma que las diferencias detectadas son reales. En el caso de las células que fueron clasificadas como no tumorales, pero aparecen clusterizadas durante el análisis *scRNA-seq* junto a otras que sí han sido identificadas como tumorales, han sido eliminadas de los análisis posteriores. Respecto al caso contrario, debido a que fueron agrupadas en función de su perfil transcriptómico junto a células no tumorales, se mantuvieron (Figura 3.7.b). Cabe destacar el clúster C9, en el que algunas de sus células aparecen representadas mediante tSNE en el espacio bidimensional de las células no tumorales. Éstas fueron eliminadas de los análisis posteriores, ya que el clúster fue identificado como tumoral y la representación bidimensional no tiene por qué ser acorde a la distribución 12-dimensional de los datos.

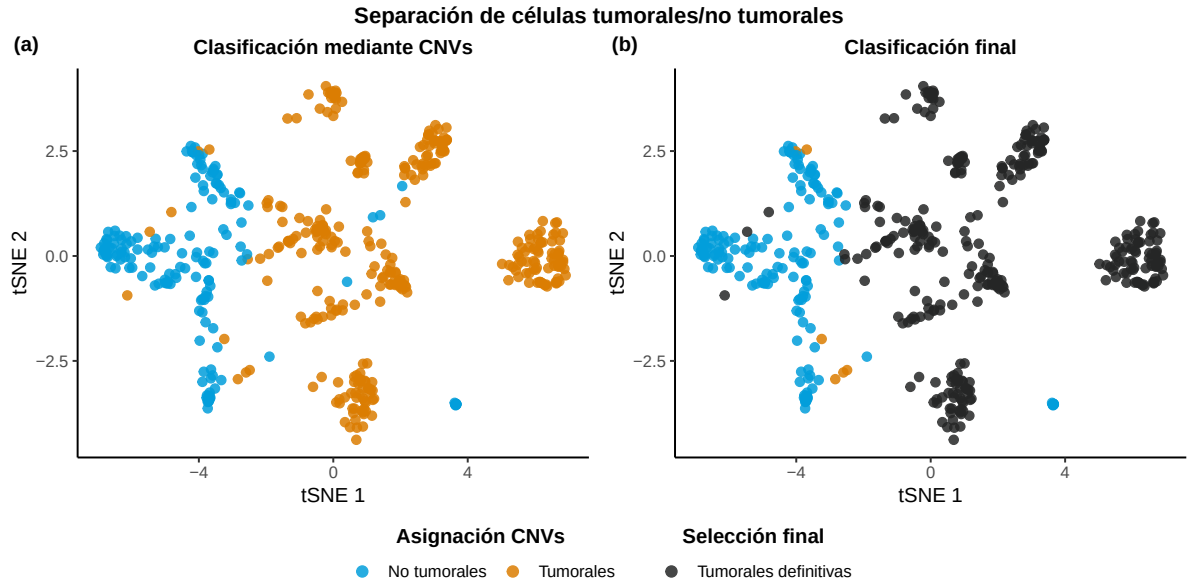

Figura 3.7: Represetación tSNE de la separación de células tumorales/no tumorales mediante CNVs. **(a):** Clasificación obtenida mediante el método basado en CNVs. **(b):** Clasificación final teniendo en cuenta los clústeres obtenidos durante el análisis *single-cell*.

3.5.2. Caracterización de células no tumorales

Tras la identificación de las células no tumorales, fueron separadas de su contraparte para ser analizadas independientemente con el fin de identificar los tipos celulares presentes entre ellas. El análisis y la preparacion de los datos consistió en los mismos pasos comentados a lo largo de la Sección 3.4, exceptuando la selección de células válidas mediante el uso de métricas de calidad.

Respecto a la normalización, se detectaron 733 genes variables sobre los 16.939 de partida utilizando un umbral de varianza residual igual a 4.2. En la reducción de la dimensionalidad, se decidió, considerando el gráfico 'de codo' y el método JackStraw, utilizar un total de 8 PCs para representar las células. Finalmente, la clusterización se llevó a cabo utilizando $k = 5$ y una resolución igual a 0.7, obteniéndose un total de 9 clústeres. Los datos fueron representados de la misma manera mediante el algoritmo tSNE, utilizando, en esta ocasión, una perplejidad igual a 20 (Figura 3.8). En este caso, debido a la eliminación de la parte tumoral del conjunto de datos, se puede observar que la distribución de las células en función de las muestras de las que proceden no presenta sesgos evidentes, ya que aparecen mezcladas entre los diferentes clústeres.

Tras la obtención de los datos normalizados y los clústeres, se procedió a la caracterización

de las células. El problema se abordó mediante dos estrategias diferentes. En primer lugar, se utilizó el paquete de R SingleR (Aran et al., 2019) para la identificación de los tipos celulares, lo cual es llevado a cabo mediante la comparación de las células con sets de datos públicos. Después, se realizó un análisis manual de marcadores específicos para confirmar los resultados y revisar los casos en los que la herramienta mostró limitaciones. A continuación, se muestran los resultados en cada caso, finalizando con la clasificación final.

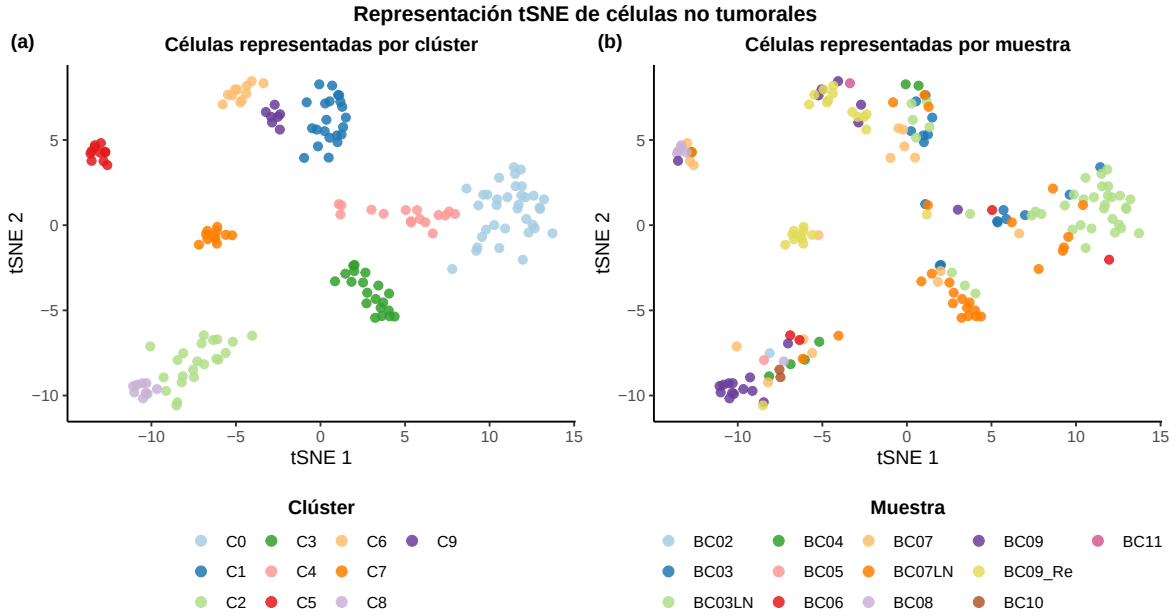

Figura 3.8: Representación tSNE de las células no tumorales. **(a):** En función de los clústeres encontrados. **(b):** En función de las muestras de las que proceden.

Clasificación mediante el uso de SingleR

SingleR es un paquete de R con el que es posible llevar a cabo la caracterización de perfiles transcriptómicos tanto a nivel celular como poblacional mediante el uso como referencia de perfiles transcriptómicos públicos, permitiendo identificar así tipos celulares canónicos. Está basado en el cálculo de correlaciones de Spearman entre los niveles de expresión de los genes más variables de las células/clústeres utilizados como entrada y las muestras utilizadas como referencia. La herramienta proporciona una puntuación a cada célula/clúster basada en el cuantil 0,8 de la distribución de correlaciones y representa la certeza de que esa célula/clúster pertenezca a un determinado tipo celular. Finalmente, lleva a cabo un paso de refinamiento en el que repite el proceso iterando sobre los tipos celulares con puntuaciones cercanas al máximo y utilizando genes marcadores de éstos. El proceso termina cuando únicamente queda una sola etiqueta para cada entrada. Como referencia, el paquete ofrece perfiles transcriptómicos procedentes de diferentes conjuntos de datos previamente caracterizados.

Respecto a este análisis, el proceso se llevó a cabo tanto a nivel de célula como a nivel de clúster. Dado que los resultados fueron consistentes entre ambos niveles, se acabó utilizando el nivel de clúster para mejorar la interpretabilidad. De entre las referencias que ofrece el paquete, se utilizaron los datos correspondientes a Blueprint + ENCODE (Martens y Stunnenberg, 2013; Dunham et al., 2012), GSE107011 (Monaco et al., 2019), Human Primary Cell Atlas (HPCA) (Mabbott et al., 2013) e ImmGen (Heng et al., 2008). En la Figura 3.9, se muestran los resultados obtenidos con las dos primeras referencias. Es posible consultar los resultados finales obtenidos con las dos últimas en la Figura B.3, así como las puntuaciones obtenidas en todos los casos antes del paso del refinamiento en la Figura B.4 en forma de mapas de calor.

En relación con los resultados, el clúster C5 obtuvo puntuaciones mayores en tipos celulares no inmunes y células indiferenciadas con todas las referencias salvo con ImmGen y GSE107011, ya que éstos únicamente presentan perfiles inmunes. Los clústeres C1, C6 y C9 fueron relacionados en todas las referencias con células T. De hecho, el clúster C6 fue clasificado como T CD8+ en 3 de las 4, mientras que C1 y C9 fueron identificados como T CD4+. Los clústeres C0 y C4 fueron clasificados en todas las referencias como células B. El clúster C3 también fue clasificado como linfocitos B por 3 de las 4 referencias, aunque cabe destacar que, en la referencia en desacuerdo (HPCA), las mayores puntuaciones las presenta en tipos celulares relacionados con ellos. Los clústeres C2 y C8 se asignaron a células con un perfil similar a macrófagos/monocitos en todos los casos. Respecto al clúster C7, los resultados fueron contradictorios, aunque, en base a las puntuaciones, parece presentar un perfil semejante a células dendríticas.

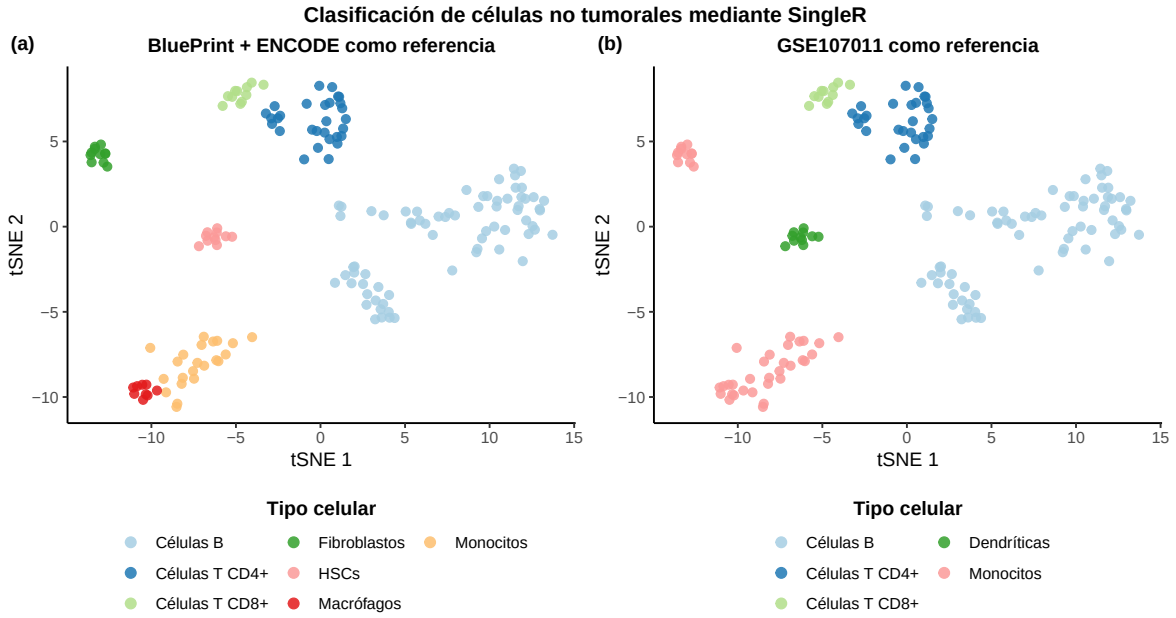

Figura 3.9: Representación tSNE de la clasificación determinada con SingleR de los clústeres de las células no tumorales. **(a):** Utilizando BluePrint + ENCODE como referencia. **(b):** Utilizando GSE107011 como referencia.

Análisis manual de marcadores específicos

Gracias a los resultados ofrecidos por SingleR, se estableció un contexto de partida sobre el que se continuó trabajando mediante el análisis de marcadores específicos de tipo celular. El objetivo fue validar los resultados y alcanzar un nivel de resolución mayor en caso de que fuera posible.

Al igual que mediante SingleR, los clústeres C1, C6 y C9 fueron caracterizados como células T debido a los altos niveles de CD3 y TCRs que exhibieron (Figura 3.10.a). Dentro de este grupo de clústeres, C6 fue identificado como células T CD8+ debido a los altos niveles de CD8A (Figura 3.10.b) y GZMB (Figura B.5.a), mientras que las células de los otros dos clústeres presentaron niveles más elevados de CD4 (Figura 3.10.c). Respecto a una caracterización a mayor resolución de los subtipos de linfocitos T CD4+, finalmente se estableció la etiqueta de células T CD4+ reguladoras (TCD4reg) al clúster C9, mientras que C1 fue determinado como células T CD4+ de memoria (TCD4mem). Respecto al clúster C9, la decisión se basó en la observación de niveles de expresión más elevados en genes relacionados con la diferenciación hacia células de memoria, como CCR7, ITGAE (Figura B.5.b y B.5.c) u OXPHOX (Figura 3.10.d), entre otros (Chang et al., 2014; Yu et al., 2020). Respecto al clúster C1, se utilizaron los resultados ofrecidos por

SingleR con la referencia HPCA (Figura B.3.b y Figura B.4.a) y la detección de niveles de expresión medios más elevados de algunos marcadores de células TCD4reg, como FOXP3 o CTLA4 (Figura B.7). Cabe destacar que el estatus funcional detectado en las células de estos clústeres fue muy variado, por lo que es posible la presencia de otros subtipos de células T CD4+ que, por la relativa baja cantidad de células, dan lugar a una señal confusa imperceptible durante la clusterización.

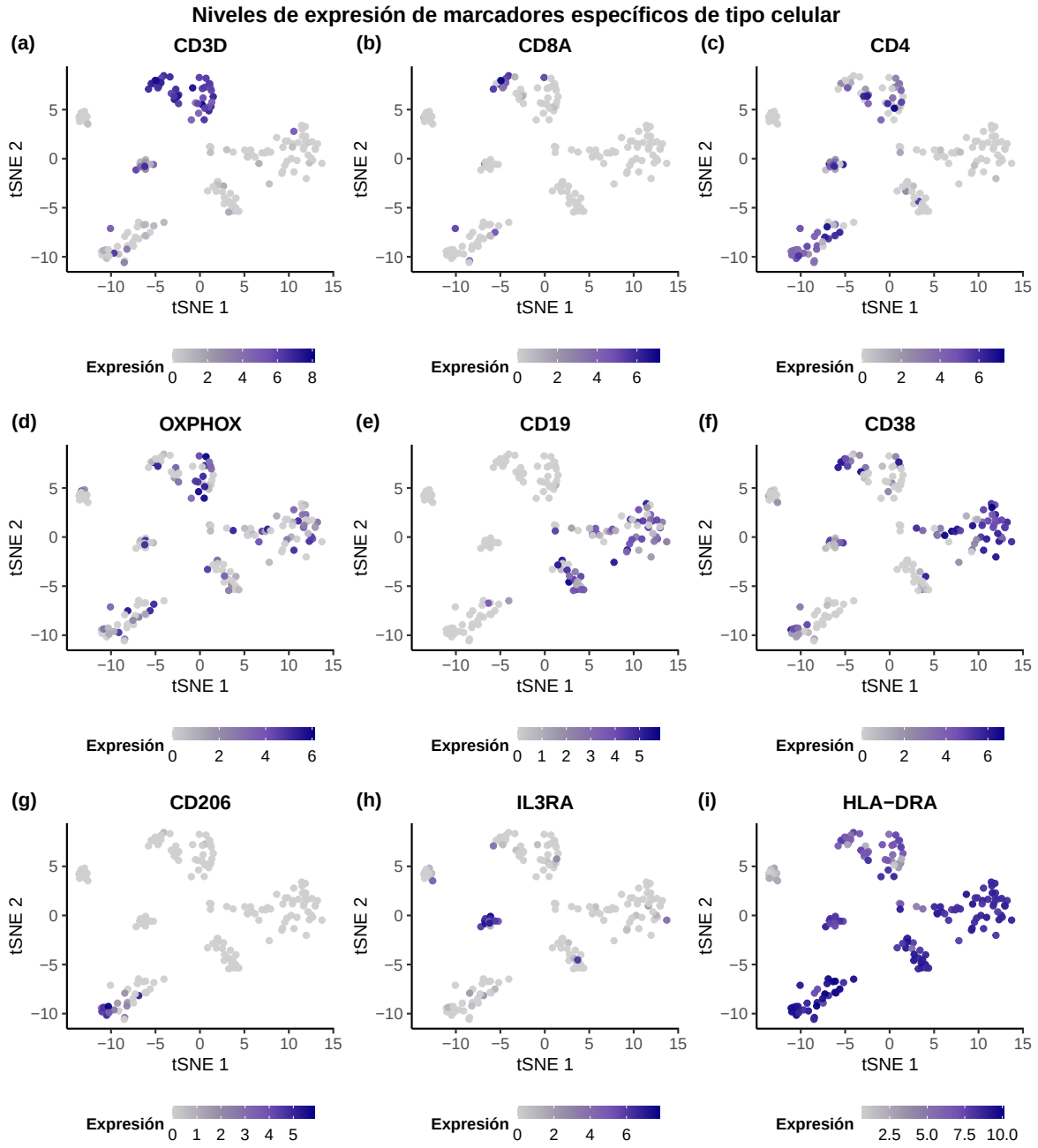

Figura 3.10: Representación tSNE de los niveles de expresión de los marcadores utilizados para la caracterización manual de las células no tumorales. El título de cada gráfico se corresponde con el gen que se está representando.

Los clústeres C0, C4 y C3 fueron clasificados, al igual que mediante SingleR, como células B por la elevada expresión de CD19 (Figura 3.10.e), CD20 (Figura B.5.f) e inmunoglobulinas. Particularmente, C0 y C4 fueron identificados como células B de centros germinales (BGC) debido a la presencia de niveles elevados de CR2, MKI67, CD10 (Figuras B.5.g-B.5.i) y CD38 (Figura 3.10.f) y a la ausencia de A4GALT (Figura B.5.j). Debido a la heterogeneidad en los

niveles de expresión y al hecho de que estas células fueron separadas en dos grupos durante la clusterización, es probable que se correspondan con diferentes estados madurativos de células BGC. Finalmente, el clúster C3 se determinó como células B de memoria maduras (Bmem) debido a la ausencia de CD38 (Figura 3.10.f) y CD10 (Figura B.5.i) y a niveles elevados de IgM, TNFRSF17, CCR6 e ITGAX (Figuras B.5.k, B.5.l, B.6.a y B.6.b) (Golinski et al., 2020; Klein et al., 2003).

En el caso de los clústeres C2 y C8, dado que SingleR los identificó como células similares a macrófagos/monocitos con todas las referencias, se analizaron marcadores específicos para poder separarlos con seguridad. Respecto al clúster C8, al igual que SingleR (Figura 3.9), fue identificado como población de macrófagos debido a los elevados niveles de CD68, CCL2, CCL5, CD163, CD16, FCGR1A, FCGR2A (Figuras B.6.c-B.6.i) y CD206 (Figura 3.10.g) (van den Bosch et al., 2017; Ruiz-Alcaraz et al., 2018; Yang et al., 2014), siendo muchos de ellos marcadores de células fagocíticas mononucleares maduras. El clúster C2 se consideró como monocitos debido tanto a los resultados de SingleR (Figura 3.9) como a la presencia de CD1 (Figuras B.6.j y B.6.k) (Kasinrerk et al., 1993). En cualquier caso, todo parece indicar que las células que componen estos clústeres se encuentran en diferentes estados madurativos del linaje de las células fagocíticas mononucleares.

Por último, el clúster C7 fue identificado como células dendríticas tanto por SingleR usando la referencia GSE107011 (Figura 3.9.b) como por los altos niveles de expresión de IL3RA (Figura 3.10.h) y CD303 (Figura B.6.l) (Boiocchi et al., 2013). El clúster C5, debido a la ausencia de marcadores de células inmunes maduras como HLA-DRA (Figura 3.10.i) y a los resultados obtenidos con SingleR (Figura 3.9.a), fue determinado como células estromales, entre las que probablemente se incluyan fibroblastos y endotelio.

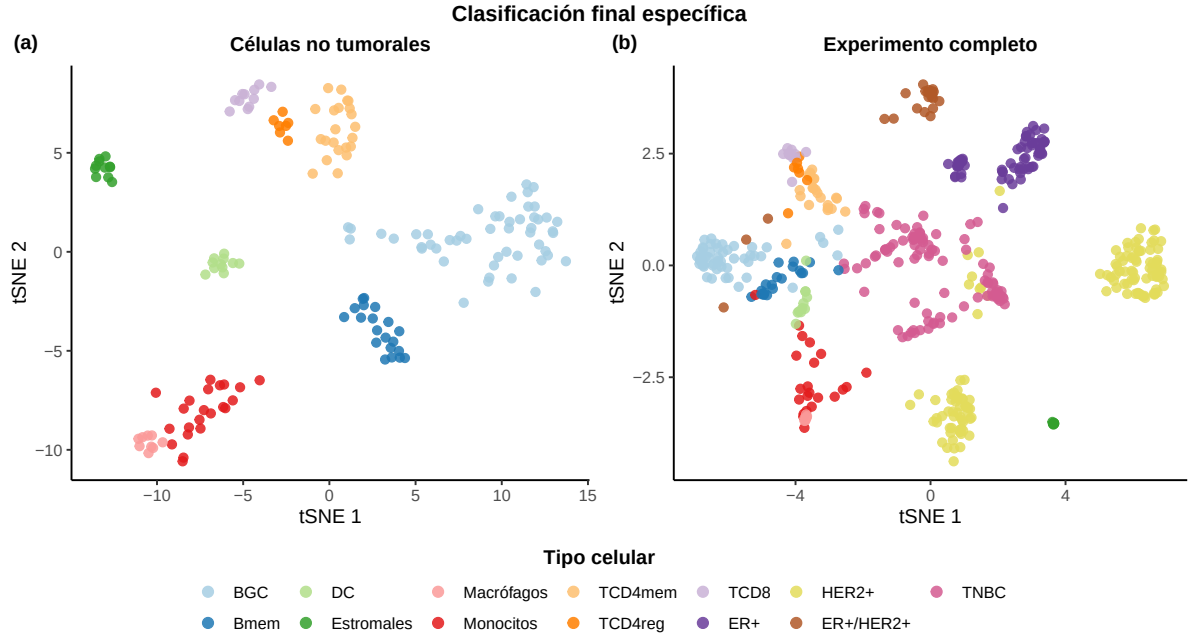

Figura 3.11: Representación tSNE de la clasificación final específica. **(a):** Únicamente células no tumorales. **(b):** Conjunto completo de células del experimento.

Es importante tener en cuenta que las clasificaciones se han establecido en base a la premisa de que cada clúster representa una agrupación de células del mismo tipo, pero que pueden presentar estados funcionales diferentes. La clasificación definitiva se muestra en la Figura 3.11, así como su representación sobre el tSNE generado con todas las células del experimento. Respecto a las células tumorales, se ha utilizado la clasificación de los subtipos intrínsecos del cáncer proporcionada por Chung et al. (2017). Estos resultados demuestran la amplia variedad de tipos

celulares presentes en el contexto tumoral, poniendo de manifiesto la necesidad de herramientas que cuantifiquen dicha heterogeneidad en muestras *bulk RNA-seq*.

Finalmente, de cara a la generación de los modelos de deconvolución, se estableció una clasificación final alternativa con un nivel de resolución menor (Figura 3.12). En concreto, monocitos y macrófagos fueron englobados en la categoría 'Monocitos', los subtipos tumorales en 'Tumor', las células T CD4+ de memoria y reguladoras en 'Células T CD4+' y las células B de centros germinales y de memoria en 'Células B', manteniéndose las células dendríticas, las T CD8+ y las estromales. De esta forma, se establecieron 7 tipos celulares en lugar de los 13 originales. El objetivo final del establecimiento de dos clasificaciones fue el uso de ambas para la construcción de dos modelos de deconvolución para cáncer de mama, uno específico, cuyos resultados consistan en tipos celulares concretos, y otro más general, en el que las proporciones consistan en agrupaciones de los anteriores.

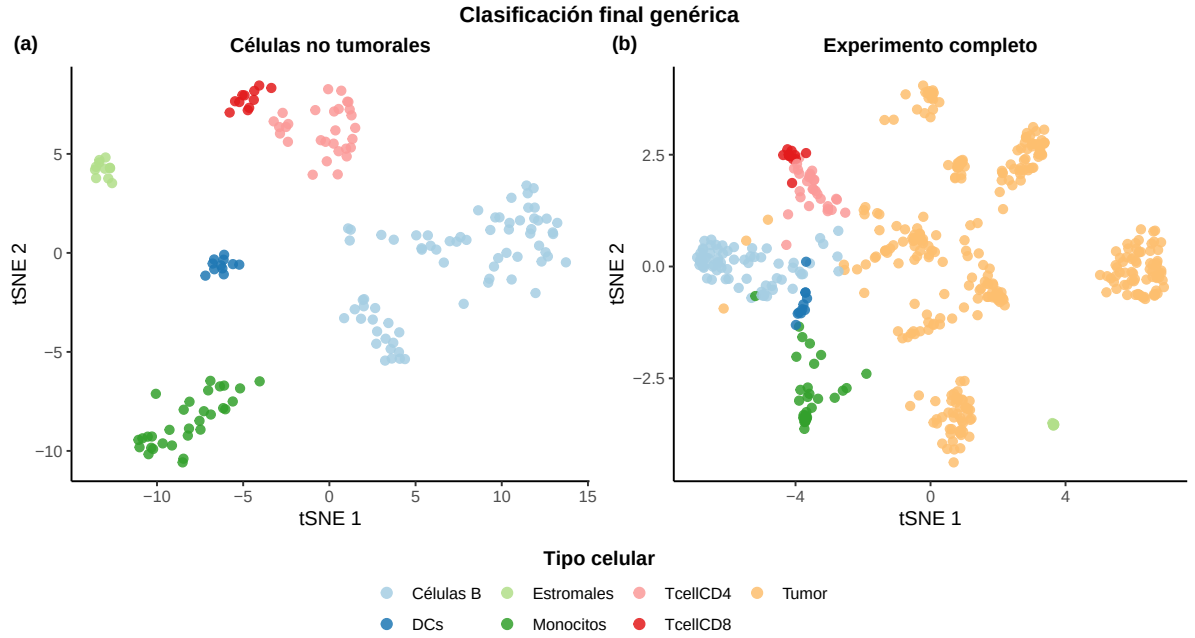

Figura 3.12: Representación tSNE de la clasificación final genérica. **(a):** Únicamente células no tumorales. **(b):** Conjunto completo de células del experimento.

4

Construcción de un modelo de deconvolución para cáncer de mama

4.1. Introducción

Tras la caracterización de las células del conjunto de datos de estudio, se utilizaron para la puesta en práctica del paquete digitalDLSorteR. El amplio repertorio de tipos celulares identificados hace que estos datos sean ideales para su uso como entrada en la herramienta. A lo largo del presente capítulo, se muestran los resultados obtenidos durante la construcción del modelo en cada uno de los pasos comentados en el Capítulo 2, demostrando no solo el funcionamiento del paquete implementado, sino también el alcance del modelo como método de deconvolución. Los modelos de deconvolución resultantes, el modelo específico y el modelo genérico, fueron integrados en el paquete en el directorio digitalDLSorteR/data para su uso como modelos pre-entrenados para la deconvolución de muestras de cáncer de mama. Además, se ha hecho una pequeña comparativa del poder de deconvolución de la red neuronal en función de los datos con los que es entrenada: únicamente las muestras *bulk*, únicamente los perfiles *single-cell* y una combinación de ambos. Finalmente, con el fin de analizar el comportamiento de los modelos sobre un conjunto de muestras *bulk* reales, se ha llevado a cabo la deconvolución de muestras de cáncer de mama procedentes del proyecto TCGA (Koboldt et al., 2012; Ciriello et al., 2015). Los resultados han sido analizados e interpretados mediante las correlaciones que se establecen entre las proporciones predichas, ya que no se cuentan con las proporciones reales de las muestras.

4.2. Simulación de nuevos perfiles *scRNA-seq*

En primer lugar, es necesaria la carga de los datos en un objeto *DigitalDLSorter*. Para ello, tal y como se muestra en la Code Box 2.2, se utilizó la función `loadRealSCProfiles` con los parámetros `min.counts` (número mínimo de *counts* por gen) y `min.cells` (número mínimo de células con más *counts* que `min.counts`) iguales a 1. De esta forma, los datos pasaron a estar compuestos por 488 células y 23.260 genes, que serán los tenidos en cuenta a la hora de su uso como método de deconvolución. Cabe destacar que, por defecto, tanto `loadRealSCProfiles` como `loadFinalSCProfiles` eliminan genes cuya expresión sea igual a cero en todas las células.

Después, como se ha comentado en el Capítulo 2, el segundo paso consiste en la estimación de los parámetros del modelo ZINB-WaVE en el caso de que haya una subrepresentación de algún tipo celular o el número de células iniciales sea reducido. Estas dos características son precisamente las que presenta el conjunto de datos utilizado, ya que el número de células iniciales es relativamente bajo y hay una mayoría de células tumorales frente a las no tumorales. Además, dentro del último grupo, hay tipos poco representados en comparación con otros, como es el caso de los macrófagos respecto a las células B. El objetivo es realizar un sobremuestreo de los datos que mejore el desempeño el modelo durante su entrenamiento mediante el aumento de la señal de aquellos tipos celulares minoritarios. Así, la red neuronal será capaz de aprender los patrones que caracterizan a estos tipos de células gracias a que los datos pasarán a estar equilibrados. El proceso es llevado a cabo con la función `estimateZinbwaveParams` (Code Box 2.3). Permite establecer covariables adicionales además del tipo celular para el ajuste de los parámetros, de forma que, en este caso, se seleccionó como covariable respecto a las células la muestra de la que proceden y, respecto a los genes, su longitud.

Una vez se estimaron los parámetros, se simularon los nuevos perfiles *single-cell* con la función `simSingleCellProfiles` (Code Box 2.3). Se generaron 1.000 células por tipo celular (argumento `n.cells = 1000`) en lugar de un número menor con el objetivo de, además de lo comentado anteriormente, aumentar la variabilidad de los perfiles de cada tipo celular y evitar una repetición excesiva de células durante la construcción de las muestras *bulk*. En la Tabla 4.1, se muestra el número final de células de cada modelo.

	Modelo general	**Modelo específico**
Número células iniciales	488	488
Número tipos celulares	7	13
Número células simuladas	7.000	13.000
Número total de perfiles	7.488	13.488

Tabla 4.1: Número de perfiles *single-cell* simulados en cada modelo.

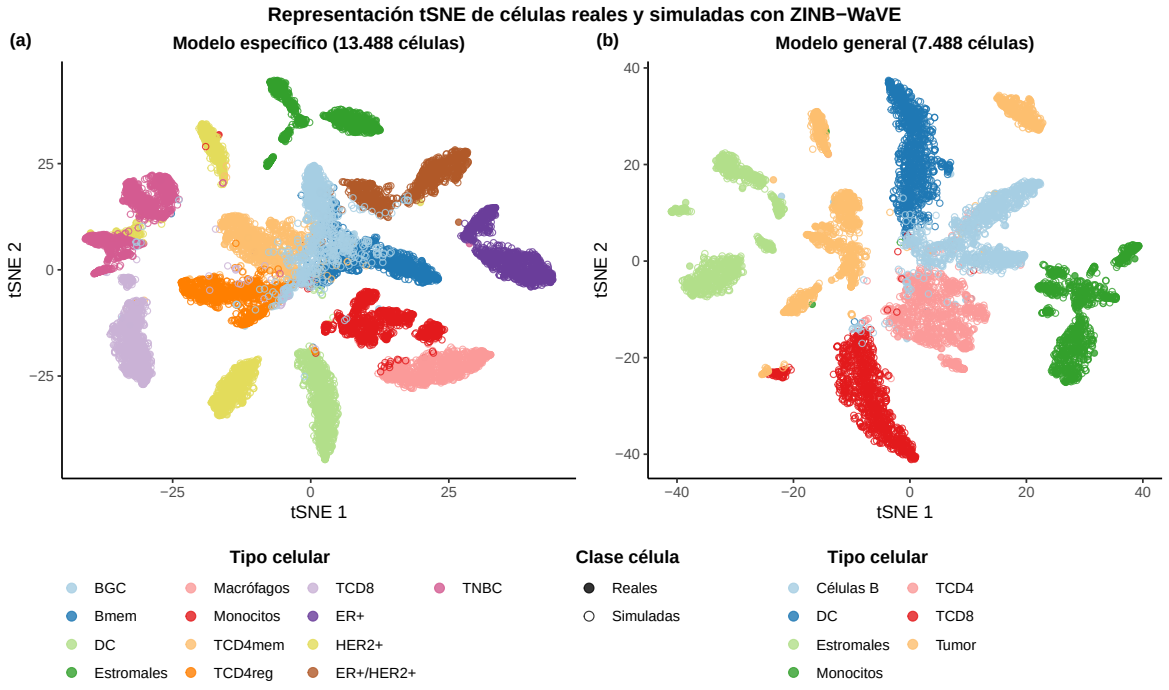

Figura 4.1: Representación tSNE de células reales y simuladas en cada uno de los modelos. **(a):** Modelo específico con 13 tipos celulares. **(b):** Modelo genérico con 7 tipos celulares.

Finalmente, para determinar si los nuevos perfiles simulados fueron coherentes con los perfiles reales y con el tipo celular al que pertenecían, se analizó el conjunto completo de células reales y simuladas mediante el procedimiento expuesto en el Capítulo 3, exceptuando los pasos de filtrado de células válidas y de clusterización. Como se puede observar en la Figura 4.1, las células simuladas colocalizan con las células reales en función del tipo celular al que pertenecen. En el caso del modelo específico, sí se puede observar que los clústeres de células T se localizan cerca unos de otros en el espacio bidimensional definido por tSNE. Este hecho es completamente lógico, ya que se tratan de perfiles relativamente similares. En cualquier caso, cada tipo celular define su propio espacio bidimensional, lo que demuestra que la simulación funciona correctamente.

4.3. Generación de matrices de composición celular y simulación de perfiles *bulk RNA-seq*

Una vez se obtuvieron los perfiles *single-cell* finales, el siguiente paso consistió en la construcción de las matrices de composición celular. Para ello, como se comenta en el Capítulo 2, digitalDLSorteR utiliza 5 métodos diferentes con el fin de evitar la existencia de sesgos durante el entrenamiento. Además, es posible la determinación de la proporción de muestras *bulk* que serán generadas de una forma u otra.

Respecto al presente análisis, dado que `generateTrainAndTestBulkProbMatrix` necesita la especificación de un rango de proporciones previas de cada tipo celular (argumento `prob.design`, Code Box 2.4), se establecieron los rangos especificados en la Tabla 4.2 para el modelo específico y general. Respecto a la proporción de muestras *bulk* generadas por un método u otro, se establecieron las proporciones por defecto en ambos conjuntos de datos, entrenamiento y test. Consisten en una mayoría de muestras con mayor aleatoriedad (85 %) y un restante mediante el muestreo aleatorio de una distribución uniforme limitada por los rangos establecidos en el argumento `prob.design` (15 %).

Tipos celulares	Desde	Hasta
ER+	30	70
HER2+	30	70
ER+/HER2+	30	70
TNBC	30	70
Estromales	1	50
Monocitos	0	15
Macrófagos	0	15
DC	0	15
TCD4mem	0	15
TCD4reg	0	15
TCD8	0	15
BGC	0	15
Bmem	0	15

Tipos celulares	Desde	Hasta
Tumor	30	70
Estromales	1	50
Monocitos	0	15
DC	0	15
TCD4	0	15
TCD8	0	15
Células B	0	15

Tabla 4.2: Rangos previos especificados para cada tipo celular. A la izquierda, para el modelo específico. A la derecha, para el modelo genérico.

En la Figura 4.2, se muestran los diagramas de caja correspondientes a la distribución de las proporciones de cada tipo celular en ambos modelos en función del método por el que han sido generadas, así como el número de muestras que serán simuladas siguiendo cada distribución. Únicamente se presenta el conjunto de datos de entrenamiento, pero las distribuciones son

equivalentes en el caso de los datos de test. El método 1 da lugar a proporciones muy similares a las especificadas mediante `prob-design`, mientras que, el resto, introducen mayor aleatoriedad. Respecto a las Figuras 4.2.a.6 y 4.2.b.6, representan la distribución de las proporciones ordenadas decrecientemente en función del número de tipos celulares considerados. Se puede observar que las muestras simuladas tuvieron una composición variada, pudiendo presentar desde un solo tipo celular hasta el total de tipos del modelo. Esta información es accesible en el paquete mediante la función `showProbPlot`.

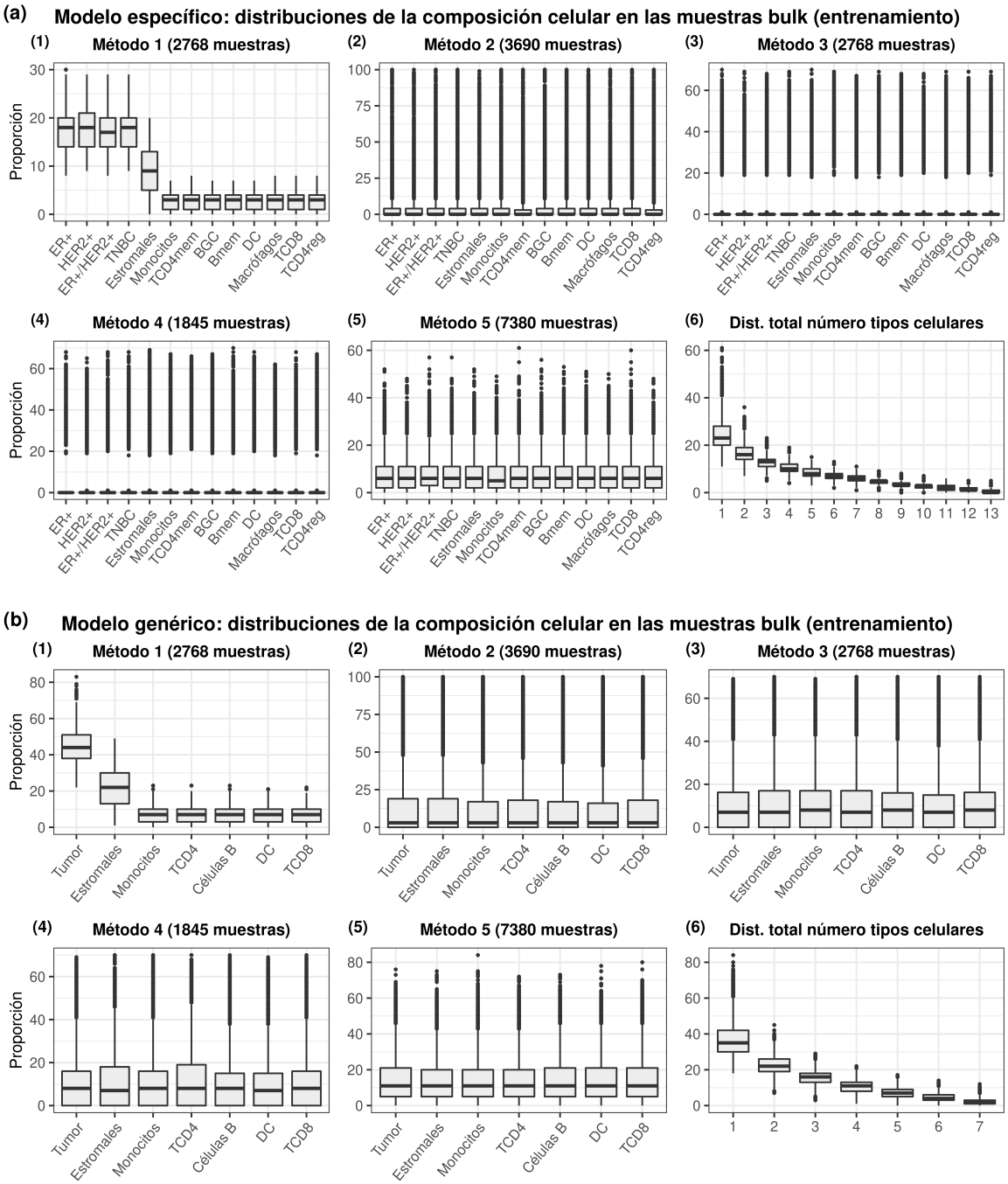

Figura 4.2: Diagramas de caja de las proporciones celulares generadas por cada método (conjunto de entrenamiento). **(a):** Modelo específico. **(b):** Modelo general. Nota: en cada apartado, los cinco primeros diagramas de caja (1-5) se corresponden con las distribuciones en función del tipo celular. Por el contrario, el número 6 es la distribución de las proporciones ordenadas decrecientemente en función del número de tipos celulares considerados.

En este paso no solo se establecen las proporciones celulares, sino también el número de muestras *bulk* que serán simuladas. En este caso, se estableció un total de 31.000 muestras para ambos modelos, de las cuales el paquete utiliza un 60 % para el entrenamiento de la red (18.451) y el 40 % restante para su evaluación (12.549). Es importante tener en cuenta que las células utilizadas para la generación de cada conjunto de muestras *bulk* son diferentes, evitando así el falseamiento de los resultados.

Con la matriz de composición celular construida, el siguiente paso consistió en la simulación de los perfiles *bulk RNA-seq* mediante la agregación de las células de acuerdo a la Ecuación 2.1 con la función `generateBulkSamples`. Después, con la función `prepareDataForTraining`, los datos son preparados para el entrenamiento de la red neuronal (Code Box 2.5). Con el fin de hacer una comparativa de la precisión de los modelos en función del tipo de datos con los que son entrenados, en el presente análisis se construyeron 3 modelos con los datos caracterizados a nivel específico: uno entrenado con únicamente los perfiles *bulk* (modelo específico 1), otro con únicamente los perfiles *single-cell* (modelo específico 2) y otro con la combinación de ambos (modelo específico 3). Esto es posible mediante el uso del argumento `combine`. Es importante tener en cuenta que los 3 modelos presentan tanto en entrenamiento como en test los mismos perfiles *bulk* y *single-cell*, independientemente de que en un modelo sean utilizados para el entrenamiento y en otro no. Ello hará más comparables los resultados, evitando posibles diferencias debido a la variabilidad de las células. Respecto al modelo genérico, únicamente se generó un único modelo entrenado con muestras *bulk*. En la Tabla 4.3, se puede encontrar el número de perfiles de cada tipo utilizado para entrenamiento y evaluación en cada modelo.

	Entrenamiento		**Test**	
Tipo de perfiles	*Bulk*	*sc*	*Bulk*	*sc*
Modelo específico 1 (*bulk*)	18.451	0	12.549	4.496
Modelo específico 2 (*sc*)	0	8.992	12.549	4.496
Modelo específico 3 (*bulk* + *sc*)	18.451	8.992	12.549	4.496
Modelo genérico (*bulk*)	18.451	0	12.549	2.496

Tabla 4.3: Número de perfiles *bulk* y *single-cell* (*sc*) utilizados para el entrenamiento y la evaluación de cada modelo.

4.4. Entrenamiento de la Red Neuronal Profunda

El entrenamiento de la red neuronal mediante la función `trainDigitalDLSorterModel` (Code Box 2.6) se llevó a cabo estableciendo un tamaño de *batch* de 128 (argumento `batch.size = 128`) y un total de 25 épocas (argumento `num.epochs = 25`) para todos los modelos excepto el genérico, donde se establecieron 20 épocas para evitar problemas de sobreajuste debido a que el número de tipos celulares es menor. El resto de parámetros fueron los establecidos por defecto, utilizando, por tanto, KLD como función de pérdida y la medición de las métricas precisión y error medio absoluto (MAE) durante el entrenamiento de la red. Los resultados finales de los modelos construidos se muestran en la Tabla 4.4. La evolución de las métricas en el caso del modelo específico entrenado únicamente con muestras *bulk* y el modelo genérico, se presentan en la Figura 4.3.

Respecto a los modelos entrenados únicamente con muestras *bulk* (modelo específico 1 y modelo genérico), se observaron resultados muy similares entre el conjunto de datos de entrenamiento y de test respecto a las métricas KLD y MAE, encontrando valores más altos en el caso del modelo específico debido a que el mayor número de tipos celulares presentes complica el problema. Sin embargo, llaman la atención los resultados claramente superiores sobre los datos de

test en el caso de la precisión, una métrica para tareas de clasificación. Este hecho se debe a que el conjunto de datos de test no solo contiene los perfiles *bulk*, sino también perfiles *single-cell*. La red aprende a caracterizar los tipos celulares mediante la búsqueda de patrones en los perfiles de expresión, por lo que las células son mucho más fáciles de clasificar correctamente que las mezclas heterogéneas debido a la ausencia de señales de otros tipos celulares que introduzcan confusión. Dado que la precisión únicamente tiene en cuenta que la etiqueta con mayor probabilidad asignada sea la correcta, el resultado son valores más elevados. Las anteriores métricas, en cambio, miden el error cometido con respecto a las proporciones predichas, por ello son más similares a pesar de la presencia de perfiles *single-cell* en test.

	KLD		**Precisión**		**MAE**	
Conjunto de datos	Entr.	Test	Entr.	Test	Entr.	Test
Modelo específico 1 (*bulk*)	0,0515	0,0751	0,7442	0,885	0,015	0,014
Modelo específico 2 (*sc*)	0,0289	0,5718	1	0,6131	7,6197	0,0465
Modelo específico 3 (*bulk* + *sc*)	0,0445	0,0385	0,7953	0,8101	0,0123	0,0112
Modelo genérico (*bulk*)	0,0222	0,0154	0,9051	0,9629	0,0186	0,0112

Tabla 4.4: Resultados de los modelos construidos sobre el conjunto de datos de entrenamiento (Entr.) y de test.

El modelo específico 2 entrenado con perfiles *single-cell*, en cambio, muestra los peores resultados en todas las métricas, presentando sobreajuste tanto en KLD como en la precisión. Respecto al MAE tan elevado sobre los datos de entrenamiento, una posible justificación es que la red no fue capaz de ajustar las proporciones de los tipos celulares de entrenamiento que, por ser únicamente perfiles *single-cell*, son siempre iguales a 1 en función del tipo celular al que pertenezcan. Esto da lugar a que los errores cometidos sean muy altos en comparación con los cometidos en los datos de test, donde el hecho de que las muestras estén compuestas por mezclas de las clases permite obtener un MAE más bajo. Por ello, a pesar de que la precisión sea perfecta, el error absoluto medio cometido es cercano a la probabilidad de asignar aleatoriamente una clase entre las 13 posibles, es decir, $100/13 \sim 7{,}69$. En el Capítulo 5 se discutirá con mayor detalle, pero este hecho parece indicar que el entrenamiento en un contexto de perfiles heterogéneos fuerza al modelo a aprender a ajustar las probabilidades de las diferentes clases, lo que, en definitiva, es el problema de deconvolución que se quiere resolver. Es posible que el bajo desempeño del modelo 2 se deba al menor número de perfiles en le conjunto de entrenamiento en comparación con el resto de modelos, pero resultados no mostrados con modelos con mayor número de perfiles presentan los mismos valores. Otra posibilidad es que la arquitectura utilizada no sea la ideal para resolver el problema únicamente con perfiles *single-cell*.

Finalmente, el modelo específico 3 entrenado con la combinación de las muestras *bulk* y los perfiles *single-cell* presenta mejores resultados en comparación con el resto de modelos específicos respecto a KLD y MAE. En el caso de la precisión, se observan resultados más equilibrados entre entrenamiento y test debido a la presencia de perfiles *single-cell* también en el entrenamiento. Sin embargo, es llamativo el hecho de que los valores de precisión en el conjunto de test de este modelo (0,8101) no llegan a ser tan buenos como los del modelo 1 (0,885). Este resultado parece apoyar la idea de que el entrenamiento con únicamente perfiles *bulk* obliga a la red neuronal encontrar patrones más específicos en los niveles de expresión. Por contra, la introducción de los perfiles *single-cell* podría estar introduciendo ruido debido a la elevada variabilidad que potencialmente pueden presentan a pesar de que pertenezcan al mismo tipo debida al ruido intrínseco de estos datos. En cualquier caso, teniendo en cuenta únicamente las métricas KLD y MAE, *a priori*, parece que el modelo específico 2 es superior al 1 entrenado únicamente con muestras *bulk*, pero es necesaria una evaluación más exhaustiva de los resultados.

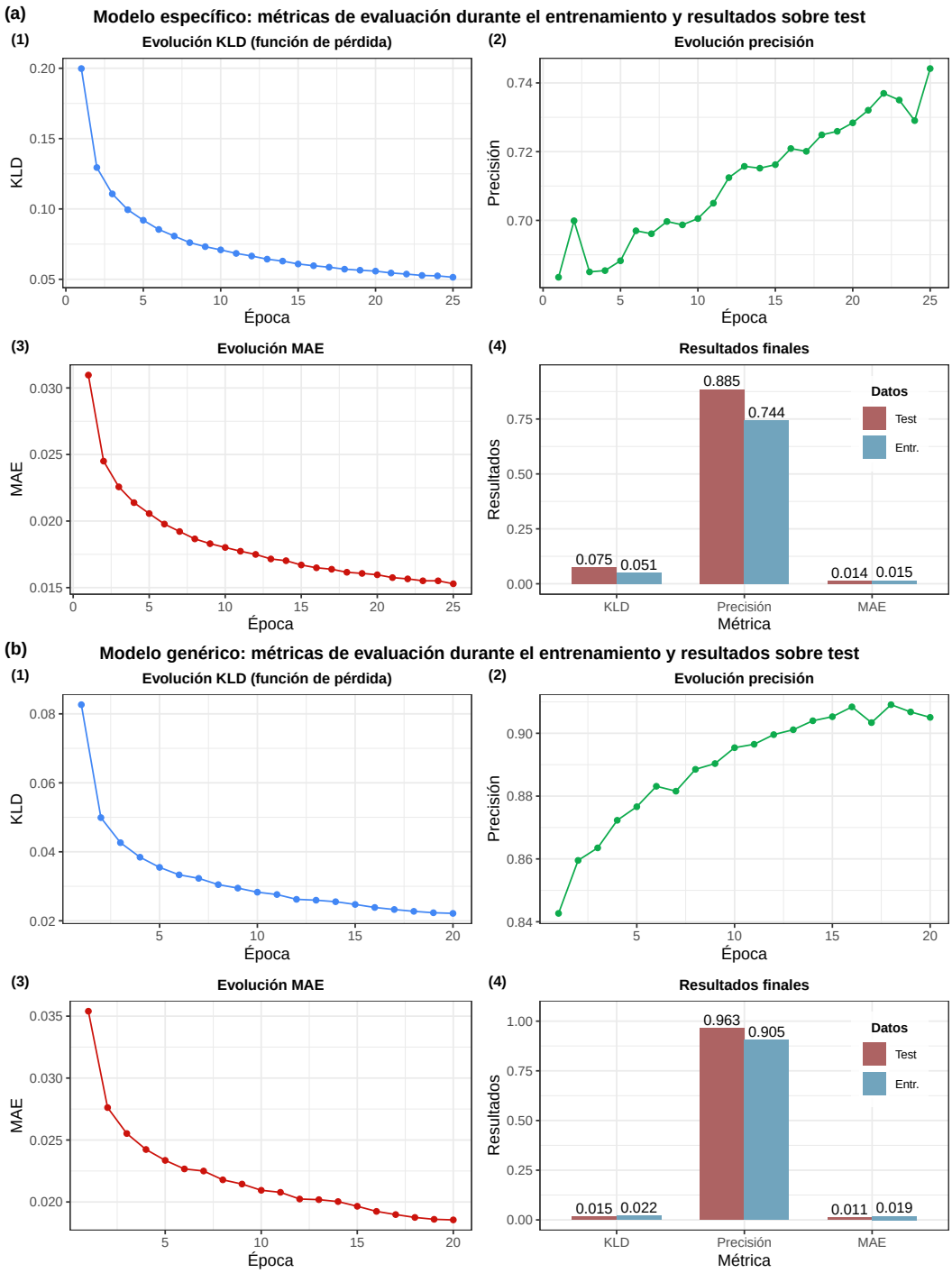

Figura 4.3: Evolución de las métricas de evaluación durante el entrenamiento del modelo (1-3) y resultados finales sobre datos de entrenamiento y test (4). **(a):** Modelo específico. **(b):** Modelo genérico.

4.5. Evaluación de los modelos sobre los datos de test

A pesar de que las métricas de evaluación durante el entrenamiento sean útiles para determinar si el modelo está siendo entrenado correctamente o no, no permiten evaluar su desempeño para la resolución del problema de deconvolución. Es interesante conocer si hay sesgos hacia tipos celulares concretos y si el grado de correlación entre las proporciones predichas y las proporciones esperadas es alto. Para ello, se llevó a cabo la evaluación de los modelos sobre los datos de test mediante la función `calculateEvalMetrics` y el uso de las funciones de visualización `distErrorPlot`, `corrExpPredPlot`, `blandAltmanLehPlot` y `barErrorPlot` (Code Box 2.7).

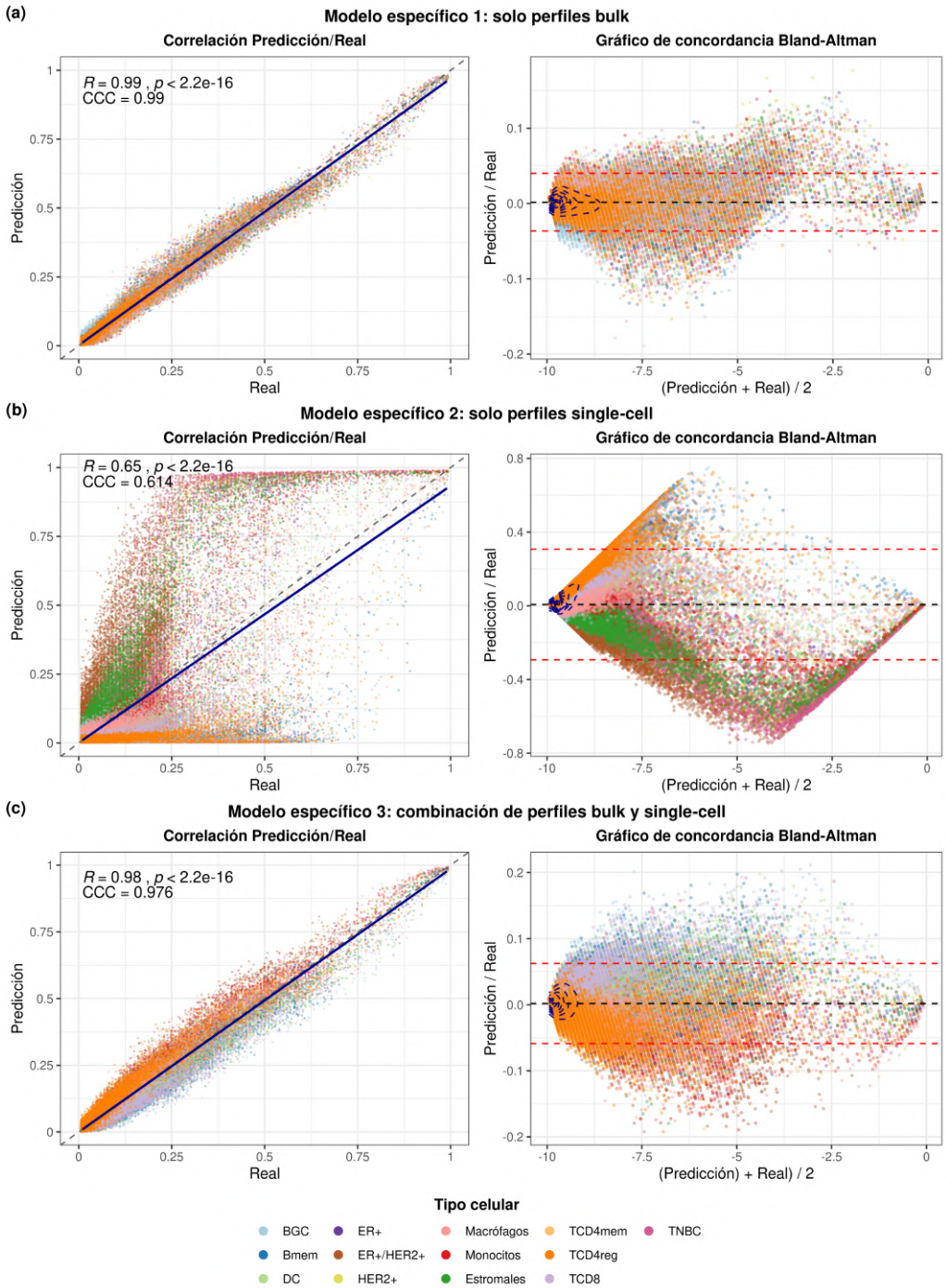

Figura 4.4: Resultados de los modelos específicos sobre el conjunto de datos de test. **(a):** Modelo específico 1 (*bulk*). **(b):** Modelo específico 2 (*single-cell*). **(c):** Modelo específico 3 (*bulk + single-cell*). A la izquierda, gráficos de correlación de las predicciones frente a las proporciones reales (línea sólida: recta ajustada; línea diagonal discontinua: identidad). A la derecha, gráficos de concordancia Bland-Altman (líneas discontinuas rojas: $\pm$ 1,96 $*$ desviación estándar sobre la media; línea discontinua negra: media; curvas discontinuas azules: densidad de puntos).

4.5.1. Comparativa de los modelos específicos entrenados con distintos tipos de datos

De la misma manera que en las métricas de error medidas durante el entrenamiento de la red neuronal, el modelo específico 2 obtuvo los peores resultados a la hora de llevar a cabo la deconvolución del conjunto de datos de test. Las predicciones con respecto a las proporciones

reales presentaron un grado de correlación de Pearson igual a 0,65 ($p < 0.001$), un CCC igual a 0,614 y umbrales de concordancia más anchos en el gráfico Bland-Altman (Figura 4.4.a). Respecto a los modelos 1 y 3, los resultados fueron muy similares, observándose mayores grados de correlación y umbrales de concordancia más estrechos el caso del primero ($R = 0{,}99$ ($p < 0.001$) y CCC = 0.99; Figura 4.4.b) que en el segundo ($R = 0{,}98$ ($p < 0.001$) y CCC = 0.976; Figura 4.4.c).

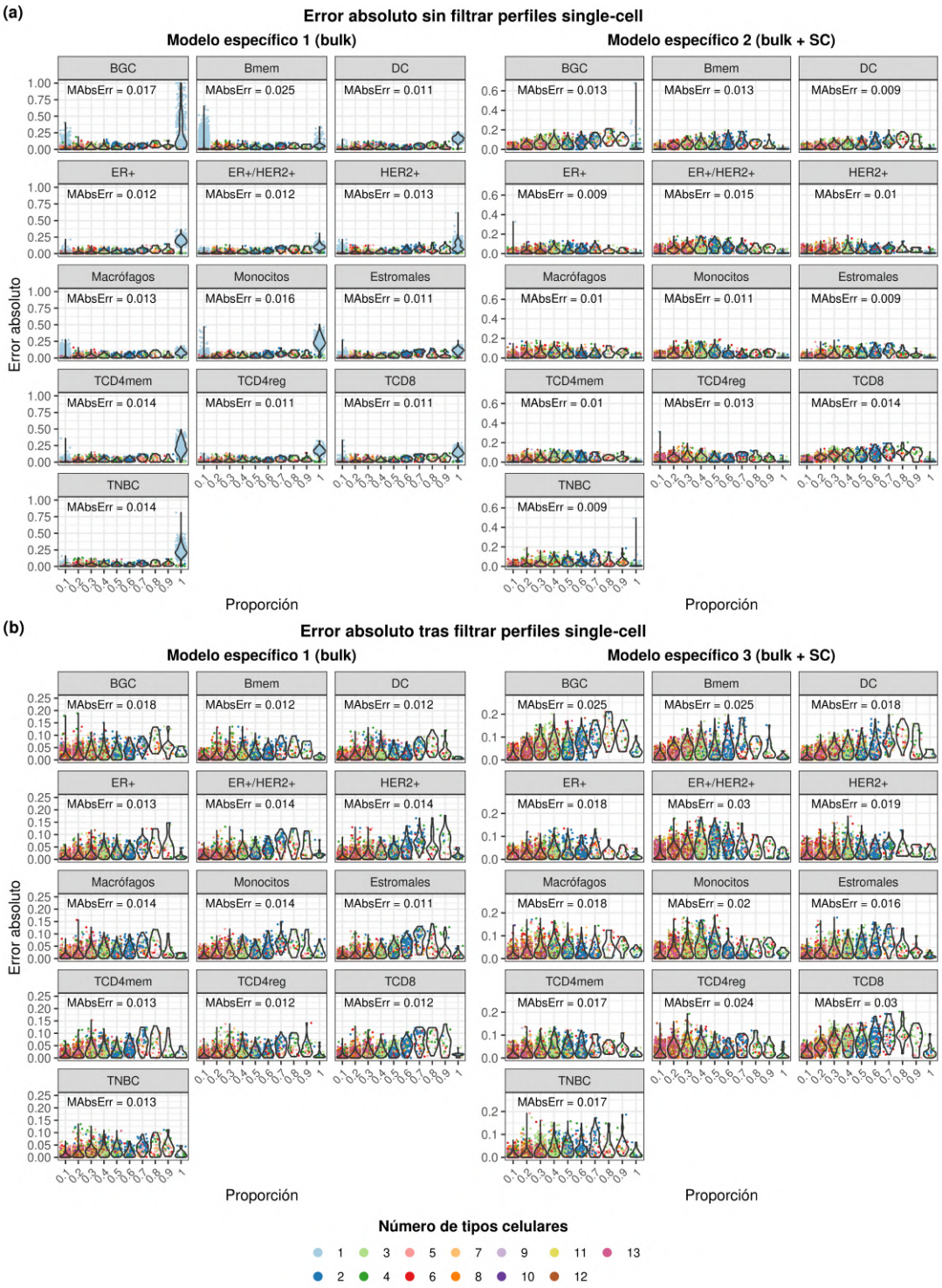

Figura 4.5: Error absoluto en función del tipo celular y por agrupaciones de proporciones de 0,1 de los modelos específicos 1 (a la izquierda) y 2 (a la derecha). **(a):** Error absoluto teniendo en cuenta tanto perfiles *single-cell* como perfiles *bulk*. **(b):** Error absoluto tras la eliminación de perfiles *single-cell*. En cada gráfico, se muestra el error medio absoluto (MAbsErr) cometido para cada tipo celular.

Para determinar las diferencias entre los modelos 1 y 3, se realizó un análisis más profundo de los errores absolutos cometidos en función del tipo celular y agrupando por rangos de 0,1 las proporciones. *A priori*, el modelo 3 presentó un mejor desempeño sobre el conjunto de datos de test completo (Figura 4.5.a). Sin embargo, mediante el filtrado de los perfiles *single-cell* para calcular el error únicamente sobre las muestras *bulk*, se observó que los resultados reportados por el modelo 1 fueron mejores a la hora de predecir la composición celular de las muestras heterogéneas (Figura 4.5.b). Esto se debe a que el modelo 2 también ha sido entrenado con perfiles *single-cell* además de los perfiles *bulk*, lo que le permite afinar más los porcentajes en situaciones de tipos celulares con proporción igual a 1 (perfiles *single-cell*). En cambio, parece que la inclusión de estos perfiles 'puros' durante el entrenamiento introduce ruido a la hora de predecir las proporciones sobre las muestras con tipos celulares mezclados, produciendo que el modelo 1 presente mejor desempeño. Cabe destacar que las diferencias medias por tipo celular son menores del 2 %, lo que demuestra que ambos modelos se comportan correctamente sobre las muestras heterogéneas. Además, a pesar de que los errores absolutos respecto a los perfiles *single-cell* sean mayores en el caso del modelo 1, las clasificaciones siguen siendo correctas como se muestra en los resultados de precisión sobre el conjunto de datos de test (Tabla 4.4), siendo incluso superiores a las del modelo 3. Esto indica que la red, a pesar de ser entrenada únicamente con perfiles *bulk*, aprende a diferenciar los patrones de expresión génica que caracterizan a cada tipo celular y es capaz de inferir a qué clase pertenecen células individuales.

4.5.2. Análisis del modelo específico

Considerando los resultados mostrados, se decidió que el modelo específico 1 entrenado con muestras *bulk* sería el incluido en el paquete como modelo preentrenado para su uso por otros usuarios. Por ello, se realizó un análisis más profundo de su rendimiento sobre los datos de test. Los resultados en función del tipo celular demuestran que el modelo funciona correctamente, presentando valores de correlación de Pearson y CCC cercanos a 0,99 en todos los casos (Figura 4.6.c). Sin embargo, puede observarse un ligero comportamiento sigmoidal en los gráficos de correlación: las predicciones llevadas a cabo entre aproximadamente el 0 y el 50 % tienden a estar sobreestimadas, mientras que aquellas entre el 50 y 75 % tienden a estar subestimadas (Figura 4.6.c). En cualquier caso, no son desviaciones alarmantes y los errores medios no superan el 2 %, indicando que, en líneas generales, el modelo funciona correctamente.

Analizando los resultados más concretamente, se puede observar que las células BGC presentan un CCC más reducido respecto al resto (CCC = 0,985), tratándose, además, del tipo celular con mayor error medio absoluto (Figura 4.6.a). Esto puede ser justificado por el hecho de que el grupo original de células B de centros germinales está compuesto por dos clústeres con un alto número de células, por lo que es probable que presente una gran variabilidad que complica la búsqueda de patrones en los perfiles de expresión génica que los identifiquen. Como se comentó en el Capítulo 3, esta variabilidad podría estar motivada por tratarse de células en diferentes estados madurativos. Se puede observar la misma tendencia en los tipos celulares tumorales, sobre todo en HER2+ y ER+/HER2+, donde los valores de CCC también son ligeramente menores que 0,99. Estos grupos presentan la misma propiedad que las células BGC, es decir, son agrupaciones de células con alta heterogeneidad, por lo que es probable que ocurra el mismo fenómeno respecto al aprendizaje de los patrones. En los resultados incluyendo los perfiles *single-cell* (Figura B.8), se puede observar cómo los valores de correlación se mantienen de forma aproximada, aunque disminuyendo sobre todo en el caso de las células BGC ($R = 0{,}96$ (p-valor $< 0{,}001$) y CCC = 0,936). Esto se debe a la subestimación de los perfiles *single-cell* comentada anteriormente (Figura 4.5) sumado al hecho de que estas células presentan una mayor variabilidad individual. Finalmente, puede observarse una tendencia a cometer errores menores conforme el número de tipos celulares aumenta (Figura 4.6.b). Se trata de un comportamiento

previsible, ya que el rango de posibles errores con respecto a las proporciones es menor conforme aumenta el número de clases. El hecho de que el error aumente en las muestras con únicamente dos tipos celulares puede tener que ver con una tendencia del modelo a asignar un mayor número de tipos celulares. Muchos de los perfiles tenidos en cuenta son relativamente similares dentro de la complejidad que presentan (células BGC y Bmem, por ejemplo). Es probable que la presencia individual de alguno de estos tipos celulares en las muestras produzca que el modelo incluya un pequeño porcentaje de otro muy similar. En cualquier caso, los errores fueron muy cercanos a cero con medianas inferiores al 0,05 % en todos los casos.

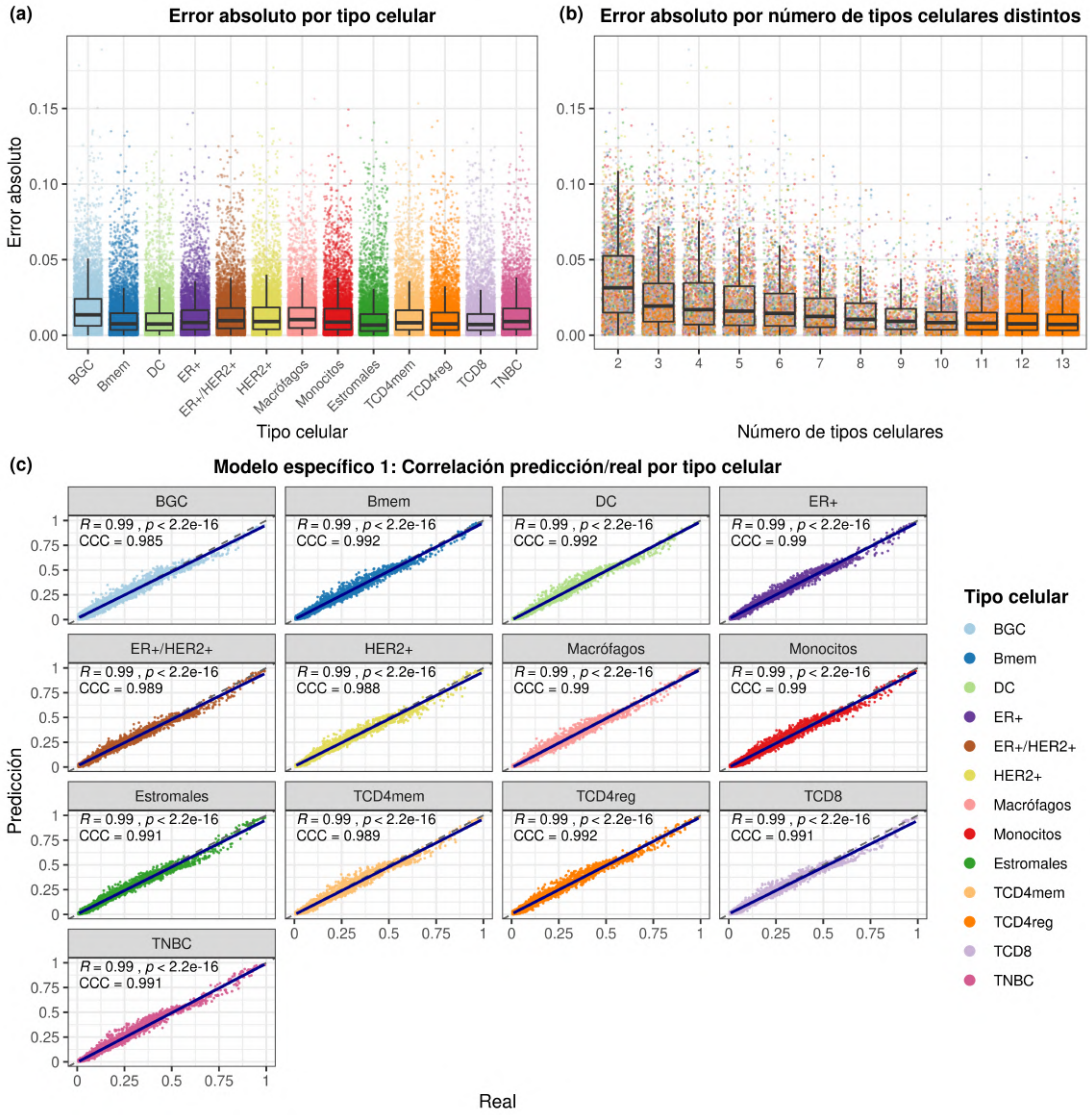

Figura 4.6: Resultados del modelo específico 1 sobre el conjunto de datos de test. **(a):** Diagramas de caja del error absoluto por tipo celular. **(b):** Diagramas de caja del error absoluto por número de tipos celulares. **(c):** Gráficos de correlación predicción/proporción real por tipo celular (línea sólida: recta ajustada; línea discontinua: identidad).

4.5.3. Análisis del modelo genérico

Teniendo en cuenta los resultados obtenidos con el modelo específico, se decidió, para la construcción del modelo genérico, utilizar únicamente los perfiles *bulk* en el entrenamiento. Los resultados obtenidos muestran que el grado de correlación linear de las proporciones predichas con respecto a las proporciones reales fue igual o superior a 0,99 ($p < 0{,}001$) en todos los

tipos celulares. Respecto al coeficiente de correlación de concordancia, todos los tipos celulares presentaron valores de 0,99 excepto las células B, con un valor de 0,988 (Figura 4.7.a). De hecho, en los errores absolutos, este tipo celular es el que presenta un valor medio más elevado (MAE = 0,019), seguido por el grupo de células tumorales (MAE = 0,016) (Figura 4.7.b). Estos resultados concuerdan con los observados en el modelo específico. Hay que tener en cuenta que, a pesar de que los datos en cada modelo (genérico y específico) hayan sido simulados independientemente por presentar distinto número de tipos celulares, los perfiles *single-cell* originales son los mismos, por lo que no es sorprendente observar las mismas tendencias en ambos casos. Respecto a los gráficos de concordancia Bland-Altman, todos los tipos celulares se ajustaron bien a los umbrales de concordancia, los cuales fueron cercanos a la media, a excepción de las células B, que presentan mayor dispersión (Figura 4.7.c).

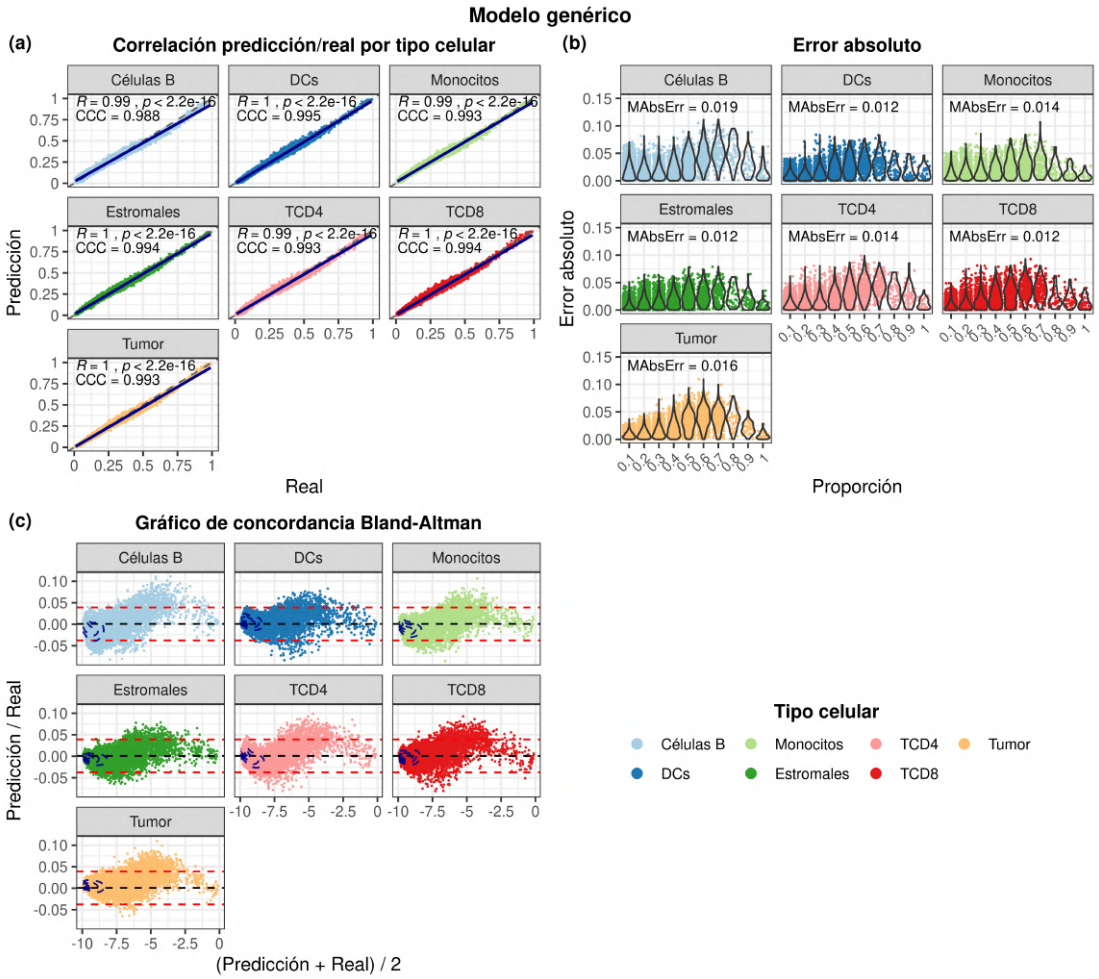

Figura 4.7: Resultados del modelo genérico sobre el conjunto de datos de test. **(a):** Gráficos de correlación predicción/proporciones reales por tipo celular (línea sólida: recta ajustada; línea discontinua: identidad). **(b):** Distribución del error absoluto por tipo celular (MAbsErr: error medio absoluto por tipo celular). **(c):** Gráficos de concordancia Bland-Altman (líneas rojas discontinuas: $\pm$ 1,96 $*$ desviación estándar sobre la media; línea discontinua negra: media; curvas discontinuas azules: densidad de puntos).

En el caso de los resultados incluyendo los perfiles *single-cell* (Figura B.9), los valores de correlación y los errores medios absolutos por tipo celular fueron muy similares, aunque se puede observar que el modelo no tiene un buen comportamiento prediciendo la proporción de los perfiles 'puros'. Por ejemplo, en el caso de los gráficos Bland-Altman (Figura B.9.c), se pueden observar umbrales de concordancia similares a los anteriores, pero dos hileras de puntos fuera de los límites que se corresponden con los perfiles *single-cell*. Por lo tanto, el comportamiento es el mismo que el observado en el modelo específico exceptuando que, debido a que el número

de perfiles *single-cell* no es tan alto como en el caso anterior (Tabla 4.3), los errores medios y las correlaciones se ven menos afectados.

4.6. Aplicación del modelo sobre datos *bulk* reales

Una vez se obtuvieron los modelos definitivos, se llevó a cabo la aplicación de ambos modelos sobre los datos públicos *bulk RNA-seq* pertenecientes al proyecto TCGA (Koboldt et al., 2012; Ciriello et al., 2015). Estos datos consisten en un total de 1.222 muestras procedentes de pacientes con cáncer de mama. Respecto al proceso de deconvolución, se realizó tal y como se muestra en la Code Box 2.8 para ambos modelos mediante la función `deconvDigitalDLSorterObj`. Dado que no se tienen las proporciones reales de los tipos celulares, no se puede estimar con precisión la validez de los resultados reportados. Sin embargo, sí se puede analizar la coherencia de las predicciones entre los diferentes tipos celulares mediante análisis de correlación entre cada par de tipos celulares. Se espera que tipos inmunes antitumorales como células CD8+ o B de memoria se correlacionen negativamente con las proporciones de los tipos tumorales, mientras que aquellas que favorecen el entorno tumoral, como los linfocitos T CD4+ reguladores, lo hagan positivamente. A continuación, se muestran los resultados para cada modelo, así como algunas consideraciones.

4.6.1. Análisis de correlación de las predicciones del modelo específico

En la Figura 4.8, se muestran los resultados de correlación obtenidos para el modelo específico. Se puede observar que los tipos celulares inmunes antitumorales como los linfocitos T CD8+ o las células B de memoria, anticorrelacionan con 3 de los 4 subtipos intrínsecos del cáncer, exceptuando ER+/HER2+. Concretamente, TCD8 presenta correlaciones de R = -0.552 para ER+ y R = -0.398 para TNBC. En cambio, en el caso de tipos celulares inmunes pro-tumorales como TCD4reg, se observa en general un alto grado de correlación con las proporciones de células neoplásicas, a excepción, de nuevo, del subtipo ER+/HER2+. Estos resultados parecen indicar que ER+/HER2+ no está siendo predicho según lo esperado. Sin embargo, de las 1.222 muestras, la proporción media predicha para este tipo celular es de 1,14 %, indicando que las correlaciones observadas presentan un rango muy estrecho. Por ello, resulta factible no tener en cuenta esta clase, ya que parece ejercer como factor de confusión para la interpretación del resto de resultados.

El modelo específico tiene en cuenta la clasificación de los subtipos del cáncer de mama como tipos celulares distintos. Esto produce que haya ocasiones en las que el modelo prediga la presencia de varios tipos tumorales en la misma muestra, lo que biológicamente no es posible. Aunque el modelo presente un buen comportamiento en estas situaciones ajustando un tipo tumoral en mayor proporción, es posible que este hecho pueda perturbar los resultados y, sobre todo, su interpretación. Por ello, se hizo uso del argumento `simplify.set` agrupando los subtipos del cáncer bajo una misma etiqueta llamada 'Tumor'. Como se observa en la Figura B.10, se detectaron las mismas tendencias comentadas en el caso anterior, pero resolviendo el problema que presentaba el grupo ER+/HER2+ cuyas proporciones predichas eran muy bajas. Se encontraron correlaciones positivas de las células tumorales con respecto a las células TCD4reg (R = 0,384), mientras que, las células T CD8+ o las B de memoria, presentaron fuertes anticorrelaciones, con R = -0,473 y R = -0,528, respectivamente.

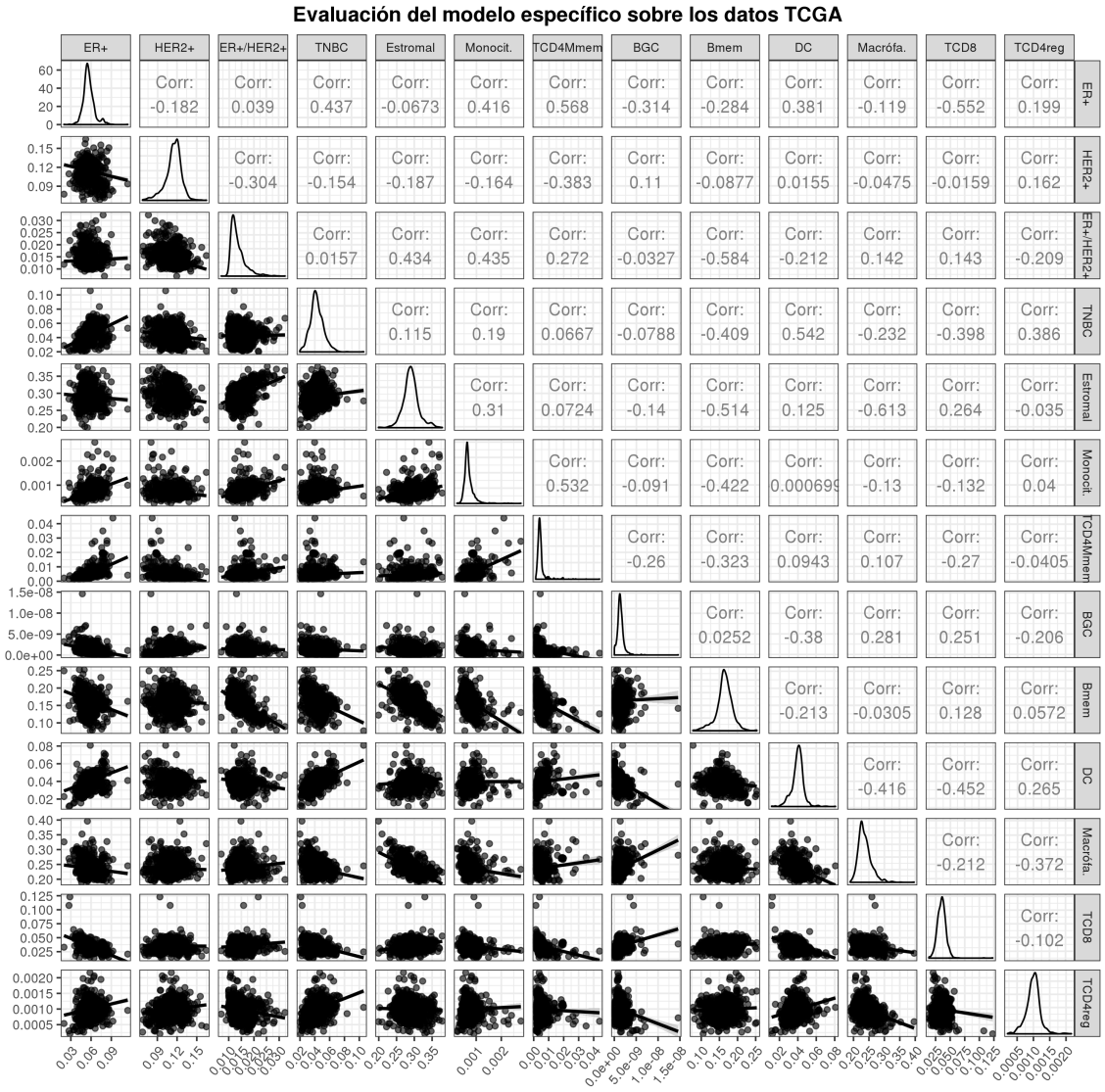

Figura 4.8: Evaluación del modelo específico sobre datos *bulk RNA-seq* reales de cáncer de mama: matriz de correlación de las predicciones de los tipos celulares considerados por el modelo.

4.6.2. Análisis de correlación de las predicciones del modelo genérico

Finalmente, se realizó el mismo procedimiento utilizando el modelo genérico. En este caso, los resultados fueron mucho más interpretables, observándose una fuerte anticorrelación de las células tumorales con los tipos celulares antitumorales, como es el caso de las células T CD8+ (R = -0,619) y las células B (R = -0,221). Por el contrario, el grupo de los monocitos, que incluye a los macrófagos y los monocitos de la clasificación anterior, se correlacionaron positivamente junto a las células TCD4+ con respecto al grupo tumoral. En el modelo específico, se observaban las mismas tendencias, siendo las TCD4reg las que presentaban una mayor correlación positiva con los tipos tumorales tanto en los resultados completos (Figura 4.8) como en los simplificados (Figura B.10). Sin embargo, es necesario destacar el caso de las células dendríticas, que en este caso correlacionan negativamente con las células tumorales, tendencia contraria a la observada en el caso anterior. Este hecho puede deberse a las inconsistencias que presenta el modelo con respecto a los tipos tumorales, lo que indica que es posible que el entrenamiento de la red en el modelo específico con todos los tipos tumorales agrupados en la clase 'Tumor' ofrecería resultados más coherentes con respecto al genérico. En cambio, sí se observa la misma correlación positiva de las células dendríticas con respecto a las células estromales en ambos modelos,

aunque más acusada en el genérico. En cualquier caso, en general, los resultados son coherentes entre ambos modelos y las correlaciones obtenidas cumplen con el comportamiento esperado considerando el papel del contexto inmune en el cáncer de mama.

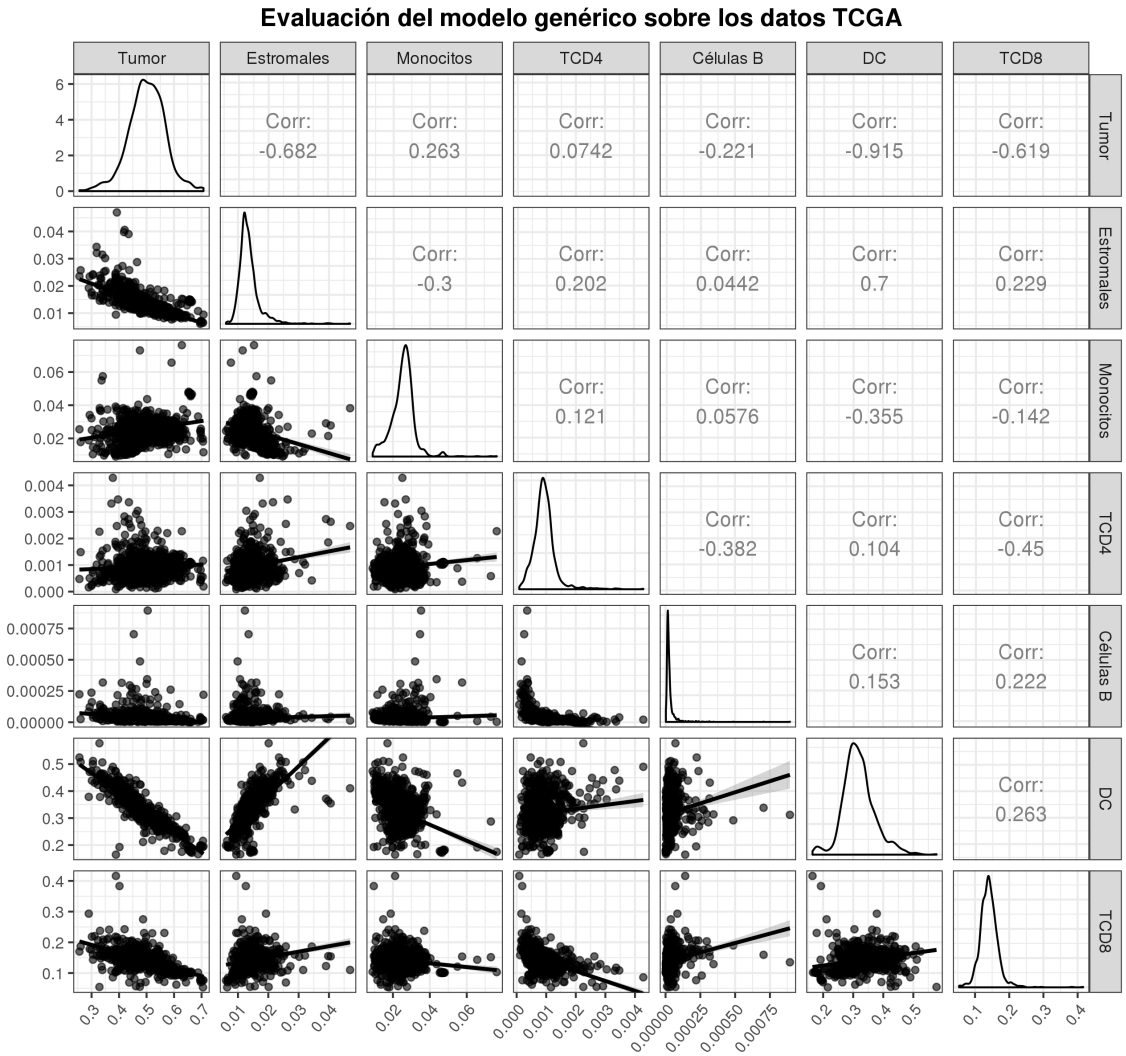

Figura 4.9: Evaluación del modelo genérico sobre datos *bulk RNA-seq* reales de cáncer de mama: matriz de correlación de las predicciones de los tipos celulares considerados por el modelo.

5

Discusión y trabajo futuro

5.1. digitalDLSorter como modelo de deconvolución

5.1.1. Aportaciones al campo de la deconvolución

Como se ha comentado en los Capítulos 1 y 3, los tejidos tumorales presentan una heterogeneidad celular que debe ser explorada por su papel clave en el progreso de la enfermedad. De hecho, gracias al incremento del número de experimentos *scRNA-seq* durante los últimos años, se han reportado evidencias cada vez mayores a favor de esta variabilidad y de sus efectos en los pacientes. Sin embargo, esta tecnología aún es cara para su aplicación de forma generalizada, aunque cada vez están apareciendo más conjuntos de datos públicos debido al potencial que exhibe. Con el fin de aportar un nuevo método para el estudio de esta diversidad celular en datos *bulk RNA-seq*, Torroja y Sanchez-Cabo (2019) propusieron digitalDLSorter, un algoritmo de deconvolución que aprovecha la valiosa información que ofrecen los datos *scRNA-seq* para resolver esta tarea. El hecho de que el punto de partida para la construcción de nuevos modelos consista en este tipo de datos tiene una serie de ventajas que, potencialmente, lo convierten en un algoritmo de deconvolución preciso.

En primer lugar, el método permite que los perfiles transcriptómicos utilizados para la simulación de perfiles *bulk* procedan directamente del entorno sobre el que se quiere construir el modelo de deconvolución. Este aspecto es fundamental, ya que, en el caso concreto del sistema inmune, los perfiles transcriptómicos de muchos tipos celulares varían en función del contexto en el que se encuentren. Sin embargo, la mayoría de métodos de deconvolución publicados utilizan como referencia perfiles de PMBCs previamente purificadas. Este hecho da lugar a una limitación que debe tenerse en cuenta, sobre todo a la hora de estimar las proporciones de tipos celulares específicos en entornos concretos, como el caso del micro-entorno tumoral en el cáncer de mama.

En segundo lugar, el incremento del número de células capturadas en los experimentos *scRNA-seq* gracias a la mejora de las tecnologías puede permitir potencialmente la caracterización transcripcional de tipos celulares no esperados. Por lo tanto, este tipo de datos permitirían alimentar al modelo no solo con tipos celulares canónicos, sino con subtipos mucho más específicos con los que, en última instancia, construir modelos más concretos y precisos. Por tanto, el desarrollo de mejores conjuntos de datos *scRNA-seq* dará lugar a modelos digitalDLSorter

superiores, produciéndose una retroalimentación positiva que favorece a ambas partes.

5.1.2. Fundamento del modelo y resultados obtenidos

Aunque mejores conjuntos de datos darán lugar a modelos de mayor calidad, la posibilidad de llevar a cabo el sobremuestreo de nuevos perfiles *single-cell* en función del tipo celular permite la obtención de modelos completamente válidos a partir de experimentos *scRNA-seq* de tamaño reducido. De hecho, a pesar del limitado tamaño del conjunto de datos utilizado en este proyecto, se ha conseguido extraer información de 9 tipos celulares inmunes distintos y la construcción de un modelo de deconvolución que, por los resultados reportados, produce buenas estimaciones. Por lo tanto, en este trabajo se demuestra que el uso del modelo ZINB-WaVE, a pesar de que supone un consumo computacional relativamente grande, permite obtener perfiles *single-cell* realistas a la hora de amplificar la señal de los datos (Sección 4.2).

En relación con la Red Neuronal Profunda, llama la atención la capacidad del modelo de, a pesar de estar compuesto por una arquitectura típicamente utilizada para problemas de clasificación, ajustar las proporciones de los distintos tipos celulares de forma precisa. Esto es posible gracias al enorme número de parámetros que presentan, siendo, en este caso, un total de 3.794.865. Además, debido a su carácter no linear, ofrecen un comportamiento robusto frente a datos ruidosos o sesgados, plano en el que los algoritmos clásicos de deconvolución basados en regresión linear presentan limitaciones. Los datos de expresión génica son relativamente ruidosos, más en el caso del *scRNA-seq*, por lo que esta propiedad, en última instancia, hará que las nuevas variables generadas durante el entrenamiento sean representaciones fieles de los tipos celulares proporcionados como entrada. Cabe destacar que, además, el protocolo no depende de la selección de marcadores previos de tipo celular como sí ocurre en la mayoría de métodos de deconvolución, ya que la selección de características es un proceso implícito en estos algoritmos. Ello facilita su usabilidad para la construcción de nuevos modelos y la deconvolución de muestras con modelos preentrenados. En cualquier caso, es probable que el filtrado de genes más variables al comienzo de la construcción de los modelos ejerza un papel positivo en los resultados posteriores.

Con respecto a los resultados reportados sobre el conjunto de datos utilizado en este proyecto y a la comparativa de los distintos modelos construidos, es destacable la necesidad de entrenar al modelo con muestras que presenten una combinación de las señales que se quieren aprender, es decir, perfiles *bulk*, para la obtención de buenas predicciones. Parece que este hecho obliga a la red a ajustar con mayor precisión las proporciones de los distintos tipos celulares y a encontrar patrones concretos en los datos de expresión génica que los caractericen. Además, la mayor similitud de estos datos con los perfiles *bulk RNA-seq* que recibirá en la práctica hace que el modelo presente un mejor desempeño. Respecto al entrenamiento únicamente con muestras *bulk* y con *bulk* y *single-cell* combinados, los modelos resultantes son prácticamente equivalentes, ya que presentan errores muy similares, aunque siendo ligeramente superior el primero. Como se ha comentado en el Capítulo 4, una posible explicación puede ser el hecho de que los perfiles *single-cell* son relativamente ruidosos con respecto a los perfiles *bulk*, ejerciendo, por tanto, un papel negativo durante el entrenamiento. Los perfiles *bulk*, por el contrario, disipan las inconsistencias existentes entre los perfiles *single-cell* por el hecho de que consisten en la agregación éstos. Sin embargo, también es importante considerar que el modelo con los datos combinados ofrece mayor robustez en caso de encontrarse con perfiles 'puros' de un tipo celular (Figura 4.5). Este aspecto quizás no sea tan relevante de cara al problema que se quiere resolver, pero puede ser un factor a tener en cuenta dependiendo del contexto de la aplicación del modelo.

Finalmente, los resultados obtenidos sobre los datos *bulk RNA-seq* reales del proyecto TCGA demuestran un buen desempeño de los modelos respecto a las correlaciones que se establecen entre las proporciones de los tipos celulares, ya que encajan con el conocimiento previo que

se tiene sobre el papel que ejercen. Cabe destacar las inconsistencias que parecen encontrarse respecto a los subtipos intrínsecos del cáncer. Parece que la separación de estos perfiles en diferentes grupos produce que las proporciones correspondientes a la fracción tumoral de las muestras sean confusas y difíciles de interpretar. Por ello, parece aconsejable la construcción de un modelo que englobe dichos tipos en uno solo, tal y como se ha realizado en el modelo genérico, pero mantenga la complejidad detectada respecto a la fracción inmune, como en el modelo específico. Los argumentos `simplify.set` y `simplify.majority` de las funciones de predicción tienen como objetivo resolver este tipo de situaciones (Figura B.10), aunque es posible que el entrenamiento directo del modelo con estos tipos celulares bajo la misma clase sea una solución más robusta. En cualquier caso, estas opciones son útiles para otro tipo de contextos, como la agregación de tipos celulares similares.

5.1.3. Limitaciones y trabajo futuro

También es importante tener en cuenta una serie de limitaciones que, en un futuro, se podrían abordar. Como cualquier proyecto, digitalDLSorter presenta varios aspectos mejorables como modelo de deconvolución.

En primer lugar, dado que se trata de un algoritmo de aprendizaje automático supervisado, es un método altamente dependiente de la calidad de los datos originales con los que es entrenado. Una caracterización incorrecta de los tipos celulares o perfiles de expresión confusos pueden hacer que el modelo disminuya considerablemente su precisión cuando se aplique sobre datos reales. En cualquier caso, este es un problema que presentan todos los métodos de deconvolución. Respecto a la cuestión planteada en relación con la variabilidad de los perfiles inmunes en diferentes entornos, es un hecho que se trata de un factor relevante a la hora de establecer patrones transcriptómicos que identifiquen a cada tipo celular. Sin embargo, debido a que los experimentos *scRNA-seq* sobre tejidos tumorales presentan un número limitado de células inmunes frente a una mayoría de células tumorales, es probable que la identificación de un mayor rango de tipos inmunes sea una tarea complicada, resultando en la imposibilidad de caracterizar todos los perfiles que potencialmente pueda haber. Por ello, podrían plantearse métodos de integración de experimentos *scRNA-seq* que permitieran incluir un mayor número de perfiles inmunes procedentes de PBMCs en el contexto del entorno tumoral capturado en el experimento *scRNA-seq* de la enfermedad. En cualquier caso, este es un problema potencialmente resoluble mediante experimentos *scRNA-seq* de mayor tamaño, tendencia cada vez mayor gracias a la mejora de las tecnologías disponibles.

Respecto al modelo en sí, es importante tener en cuenta que, a pesar de llevar a cabo su evaluación sobre un conjunto de muestras *bulk* simuladas a partir de perfiles *single-cell* independientes del conjunto de datos de entrenamiento, no dejan de proceder del mismo experimento, lo que puede dar lugar a una sobreestimación de los resultados. Para evitar este problema, una posible solución podría consistir en la construcción de un conjunto de muestras de test del mismo contexto sobre el que se construye el modelo (por ejemplo, cáncer de mama), pero procedentes de un experimento distinto. De esta forma, se obtendrían resultados más realistas para evaluar su desempeño.

En relación con el uso de Aprendizaje Profundo, a pesar del buen desempeño que ofrece, uno de los principales problemas que presenta es su naturaleza de 'caja negra' que impide que los resultados sean fácilmente interpretables. Sin embargo, existen algunos métodos que permiten resaltar aquellos patrones en las variables de entrada que son utilizados para la resolución de la tarea. Por ejemplo, en el campo de las redes neuronales convolucionales para la clasificación de imágenes, clásicamente se han utilizado los mapas de prominencia (*saliency maps*) para obtener mapas de calor de los píxeles con mayor relevancia durante el proceso. Este y otros

métodos están siendo cada vez más utilizados para la interpretación de los resultados ofrecidos por el Aprendizaje Profundo en el campo de la Bioinformática (Kimmel y Kelley, 2020; Wilentzik Müller y Gat-Viks, 2020), por lo que podría ser una buena opción para la mejora de la interpretabilidad, permitiendo saber si el modelo aprende patrones coherentes con el tipo celular en cuestión, e incluso la búsqueda de nuevos marcadores.

Finalmente, con el fin de evitar que el modelo asigne proporciones celulares confusas con porcentajes muy bajos, una posibilidad podría consistir en la inclusión de una nueva categoría denominada 'Otras células'. Representaría aquellas proporciones de la muestra cuya predicción no es confiable para el resto de tipos celulares. Respecto a su implementación en la red neuronal, el planteamiento podría constar de la simulación de un perfil transcriptómico aleatorio que debería ser incluido durante el entrenamiento. En cualquier caso, sería necesaria una evaluación del alcance de la idea, ya que podría producir que el modelo aprendiera de manera incorrecta los patrones de los tipos celulares y acabase dando lugar a la subestimación del resto de perfiles.

5.2. Implementación del modelo como paquete de R

5.2.1. Aportaciones

Como se ha descrito en el Capítulo 2, a pesar de los buenos resultados que ofrece digitalDLSorter, su implementación en forma de *pipeline* escrita en varios lenguajes de programación complicaba su uso por otros usuarios para la construcción de nuevos modelos. Además, no ofrecía métodos de evaluación de los resultados ni modelos preentrenados con los que deconvolucionar muestras directamente sin necesidad de construirlos nuevamente. Ahora, gracias a su implementación como paquete de R, se han resuelto todos estos problemas. La herramienta es más accesible por otros investigadores, se ha facilitado su uso mediante la creación de una documentación y presenta una mayor integración con el entorno típicamente utilizado para el análisis de datos transcriptómicos, que suele ser R y los paquetes de Bioconductor. Esta también es la razón por la que se han utilizado algunas de las clases que el repositorio ofrece, como *SingleCellExperiment* o *SummarizedExperiment*. Además de haber facilitado la implementación de la herramienta, permitirán que otros usuarios incorporen sus datos al objeto *DigitalDLSorter* de forma sencilla.

Hay que subrayar que digitalDLSorteR no es simplemente la introducción del código presente en la *pipeline* original en una serie de funciones. Se ha llevado a cabo la implementación de cada uno de los pasos de manera más formal, de forma que no solo presenta las ventajas ya comentadas de cara al usuario, sino también otras para el desarrollador. El hecho de haber seguido un estilo modular durante su desarrollo permite que, en un futuro, la implementación de nuevas funcionalidades o la modificación de las existentes no suponga grandes cambios en el diseño de la herramienta, haciendo del mantenimiento una labor más sencilla. Cada una de las funciones intenta trabajar de forma aislada sobre la tarea que debe llevar a cabo, lo que facilita la modificación de una parte del código sin afectar al resto. Además, se han hecho algunas mejoras respecto a la eficiencia computacional de los procesos llevados a cabo. La más destacable es la posibilidad de utilizar ficheros HDF5 como *back-end* en los pasos que tienen que ver con la simulación de las muestras *bulk*. Debido a que estas matrices no pueden representarse como dispersas y a que el número de muestras simuladas es generalmente muy alto, mantener toda esa información en memoria puede complicar los procesos posteriores, destacando el entrenamiento de la red neuronal. Ahora, es posible construir nuevos modelos sin necesidad de comenzar una nueva sesión de R tras cada uno de los pasos llevados a cabo o escribir en disco los datos intermedios en forma de ficheros tabulados. Además, el entrenamiento de la red mediante el uso de estos ficheros es más eficiente que mediante la lectura de ficheros tabulados, que era la forma

utilizada en el código original. Este formato permite un acceso veloz a las muestras gracias a que ofrece la posibilidad almacenar los datos en bloques de dimensiones deseadas. Como se ha comentado en la Subsección 2.4.2, digitalDLSorteR guarda las matrices de expresión por filas, ya que es la manera en la que son accedidas durante el entrenamiento, aumentando así la velocidad de lectura y reduciendo los tiempos de ejecución durante el entrenamiento considerablemente.

5.2.2. Limitaciones y trabajo futuro

Hay que tener en cuenta que digitalDLSorteR, por el momento, no es una herramienta lista para su publicación en un repositorio oficial, sino más bien una primera versión en la que se han implementado todas las funcionalidades y que, por supuesto, es posible utilizar (véase Material Suplementario A), ya que aprueba todas las comprobaciones realizadas por el comando `R CMD check`. Todos los resultados mostrados en el Capítulo 4 han sido desarrollados utilizando la herramienta como usuario, lo que demuestra su funcionamiento y operatividad. Sin embargo, hay que tener en cuenta algunas consideraciones y limitaciones que en este momento presenta.

La primera y más importante es la ausencia del directorio test en el paquete. El testeo y su publicación es una parte fundamental en el desarrollo de *software* de calidad, ya que es la manera en la que se demuestra el correcto funcionamiento del código implementado. Por supuesto, a lo largo del desarrollo de digitalDLSorteR se han desarrollado los tests pertinentes para cada una de las funciones que lo componen, ya que es la única manera de asegurar el correcto comportamiento de la herramienta. Sin embargo, aún no han sido formalizados mediante el uso del paquete thestthat (Wickham, 2011), la elección más común en R. Próximamente, se llevará a cabo su formalización mediante dicha herramienta y su inclusión en el repositorio del paquete.

Respecto a otros aspectos menos relevantes, pero que mejorarían la calidad de la herramienta, pueden destacarse los siguientes. Durante la simulación de nuevos perfiles *bulk*, como se ha comentado, se ha permitido el uso de ficheros HDF5 como *back-end* con el fin de ofrecer al usuario un uso de memoria RAM más óptimo durante los pasos siguientes. Sin embargo, la simulación de los perfiles sí es llevada a cabo en memoria, aunque posteriormente pasen a ser almacenadas en disco. Este hecho produce un pico en memoria correspondiente a las muestras que estén siendo simuladas en ese momento, ya sea el conjunto de entrenamiento o de test. Esto se debe a que el paquete escribe secuencialmente cada objecto tras su generación en ficheros HDF5, por lo que la memoria ocupada por el mismo es liberada. En cualquier caso, existe un pico en el uso de memoria que puede suponer un problema en el caso de simular un número de muestras superior al que pueden ser localizadas en RAM. Una posible solución consistiría en la simulación de las muestras por bloques de un tamaño determinado. Cada bloque sería añadido al fichero HDF5 de forma equivalente a como trabaja el paquete HDF5Array. Para hacer más eficiente el proceso, se podría implementar en C++ mediante los paquetes Rcpp (Eddelbuettel y Balamuta, 2017) y beachmat (Lun et al., 2018), que permiten la recepción y el manejo de matrices de R desde este lenguaje. C++ es un lenguaje compilado que presenta menores tiempos de ejecución que R, por lo que el proceso sería eficiente no solo a nivel de uso de memoria, sino también respecto a tiempos de ejecución. De esta manera, se posibilitaría la simulación de un número de muestras *bulk* mayor sin necesidad de que el *hardware* disponible sea capaz de mantenerlas en memoria. Además, pensando en la posibilidad de experimentos *scRNA-seq* mucho más grandes en los que el uso de matrices dispersas sea insuficiente, podría plantearse la implementación del uso de ficheros HDF5 también en los pasos relacionados con los perfiles *single-cell*, opción que actualmente no está disponible.

Siguiendo con la cuestión de la eficiencia computacional, respecto al entrenamiento de la red neuronal, se decidió, por lo comentado en la Subsección 2.4.2, implementar el uso de generadores que permitiesen evitar la carga completa de las matrices de expresión en memoria. Dichos generadores han sido implementados en R, proporcionando un buen desempeño en términos de

velocidad gracias a la rápida lectura que ofrecen los ficheros HDF5. Sin embargo, debido a la imposibilidad de paralelizar el código en R, la implementación actual del modelo mediante Keras no permite la paralelización del entrenamiento. Las redes neuronales tienen la ventaja de que las operaciones requeridas para su entrenamiento son fácilmente paralelizables mediante el uso de no solo de CPU (Unidad Central de Procesamiento), sino de GPU (Unidad Gráfica de Procesamiento), que presentan un mayor número de núcleos y permiten reducir considerablemente los tiempos de ejecución. Para resolver esta situación, una posible solución puede consistir en la implementación de los generadores en Python. Dado que Python sí es paralelizable y además se trata del lenguaje nativo de Keras, se podrían diseñar en este lenguaje y llevar a cabo su llamada desde R mediante el paquete reticulate (Ushey et al., 2020), una interfaz de R para Python. De hecho, es este el paquete que Keras utiliza para realizar las llamadas a Python desde R, por lo que ya se trata de una dependencia secundaria en digitalDLSorteR.

Dejando el asunto de la eficiencia y tratando el plano de las funcionalidades que ofrece, es importante tener en cuenta que el paquete implementa la misma arquitectura seleccionada por Torroja y Sanchez-Cabo (2019) sea cual sea el problema a resolver. Sin embargo, dado que el objetivo de la herramienta es ofrecer el mejor desempeño en cualquier caso, puede que el aumento en el número de tipos celulares o sencillamente una complejidad mayor en el conjunto de datos de partida produzcan que la arquitectura preestablecida no sea la ideal. En principio, dado que se espera que el tipo de datos utilizados como entrada sean similares a los que se utilizaron para el establecimiento de la arquitectura, el diseño predeterminado debería ser suficiente. En cualquier caso, para hacer más flexible su funcionamiento y ofrecer un mayor nivel de personalización, una posibilidad podría consistir en la implementación de un argumento en la función `trainDigitalDLSorterModel` mediante el cual el usuario pudiera incluir su propia arquitectura construida mediante Keras. De esta forma, sería posible la selección del modelo predeterminado o, para usuarios más expertos, la inclusión de su propia arquitectura.

Finalmente, más allá del modelo como paquete de R, se podría plantear la posibilidad de ofrecer digitalDLSorter como un servicio para la deconvolución de muestras *bulk RNA-seq* a través de una plataforma Web. Bastaría con la implementación en el lado del servidor de la parte relacionada con las predicciones, la cual sería posible realizar tanto en R como en Python por la posibilidad de utilizar Keras desde ambos lenguajes. En el caso de seleccionarse R, podría hacerse uso de Shiny (Chang et al., 2020), un paquete orientado al desarrollo de aplicaciones Web interactivas utilizando R como lenguaje de *back-end*. En cualquier caso, podría ser una alternativa al uso del paquete por usuarios que únicamente pretenden deconvolucionar sus muestras *bulk RNA-seq*, facilitando así aún más su uso.

Glosario de acrónimos

A continuación, se listan por orden de aparición todas las siglas y acrónimos utilizados en el trabajo. Aquellas entradas cuyas siglas se correspondan con la versión inglesa, son presentados en dicho idioma.

- **TILs**: *Tumor Infiltrated Lymphocytes.*
- **TNBC**: *Triple Negative Breast Cancer.*
- **NGS**: *Next-Generation Sequencing.*
- ***RNA-seq***: *Ribonucleic Acid Sequencing*
- **ARN**: Ácido ribonucleico.
- **DNN**: *Deep Neural Networks.*
- **PBMC**: *Peripheral Blood Mononuclear Cell.*
- **POO**: Programación Orientada a Objetos.
- **HDF5**: *Hierarchical Data Format 5*
- **RAM**: *Random Access Memory.*
- **SRA**: *Sequence Read Archive.*
- **IDE**: *Integrated development environment.*
- **TCGA**: *The Cancer Genome Atlas.*
- **TPM**: *Transcript per million.*
- **ZINB**: *Zero-Inflated Negative Binomial.*
- **CPM**: *Counts per million.*
- **API**: *Application Programming Interfaces.*
- **ReLU**: *Rectified Linear Unit.*
- **KLD**: *Kullback-Leibler Divergence.*
- **CCC**: Coeficiente de Correlación de Concordancia
- **GEO**: *Gene Expression Omnibus.*
- **ARNm**: Ácido ribonucleico mensajero.
- **ADNc**: Ácido desoxirribonucleico complementario.

- **FPKM**: *Fragments per kilobase million.*
- **UMI**: *Unique Molecular Identifier.*
- **PCA**: *Principal Component Analysis.*
- **kNN**: *k-Nearest Neighbors.*
- **tSNE**: *t-Distributed Stochastic Neighbor Embedding.*
- **CNV**: *Copy Number Variation.*
- **GTEx**: *Genotype-Tissue Expression.*
- **UPGMA**: *Unweighted Pair Group Method with Arithmetic Mean.*
- **HPCA**: *Human Primary Cell Atlas.*
- **MAE:** *Mean Absolute Error.*
- **CPU:** *Central Processing Unit.*
- **GPU:** *Graphics Processing Unit.*

Bibliografía

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, ..., y X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Versión 2.0.0. tensorflow.org.

B. Alberts, A. Johnson, J. Lewis, D. Morgan, M. Raff, K. Roberts, P. Walter, J. Wilson, y T. Hunt. *Biología molecular de la célula.* Omega, Barcelona, 6a ed. edition, 2016.

JJ Allaire y François Chollet. *keras: R Interface to 'Keras'*, 2020. Versión 2.3.0. www.CRAN.R-project.org/package=keras.

D. G. Altman y J. M. Bland. Measurement in medicine: the analysis of method comparison studies. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32(3):307–317, sept. 1983. doi: 10.2307/2987937.

S. Andrews, F. Krueger, A. Segonds-Pichon, L. Biggins, C. Krueger, y S. Wingett. FastQC. Babraham Institute, en. 2012. Versión 0.11.8.

D. Aran, A. P. Looney, L. Liu, E. Wu, V. Fong, A. Hsu, S. Chak, R. P. Naikawadi, P. J. Wolters, A. R. Abate, A. J. Butte, y M. Bhattacharya. Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. *Nat. Immunol.*, 20(2):163–172, feb. 2019. doi: 10.1038/s41590-018-0276-y.

F. Avila Cobos, J. Vandesompele, P. Mestdagh, y K. de Preter. Computational deconvolution of transcriptomics data from mixed cell populations. *Bioinformatics*, 34(11):1969–1979, jun. 2018. doi: 10.1093/bioinformatics/bty019.

D. Bates y M. Maechler. *Matrix: sparse and dense matrix classes and methods*, 2019. R package version 1.2-18.

E. Berglund, J. Maaskola, N. Schultz, S. Friedrich, M. Marklund, J. Bergenstråhle, F. Tarish, A. Tanoglidi, S. Vickovic, L. Larsson, F. Salmén, C. Ogris, K. Wallenborg, J. Lagergren, P. Ståhl, E. Sonnhammer, T. Helleday, y J. Lundeberg. Spatial maps of prostate cancer transcriptomes reveal an unexplored landscape of heterogeneity. *Nat Commun*, 9(1):2419, jun. 2018. doi: 10.1038/s41467-018-04724-5.

L. Bingle, N. J. Brown, y C. E. Lewis. The role of tumour-associated macrophages in tumour progression: implications for new anticancer therapies. *J. Pathol.*, 196(3):254–265, mar. 2002. doi: 10.1002/path.1027.

L. Boiocchi, S. Lonardi, W. Vermi, S. Fisogni, y F. Facchetti. BDCA-2 (CD303): a highly specific marker for normal and neoplastic plasmacytoid dendritic cells. *Blood*, 122(2):296–297, jul. 2013. doi: 10.1182/blood-2013-05-500413.

E. A. Boyle, Y. I. Li, y J. K. Pritchard. An expanded view of complex traits: from polygenic to omnigenic. *Cell*, 169(7):1177–1186, jun. 2017. doi: 10.1016/j.cell.2017.05.038.

L. J. Carithers, K. Ardlie, M. Barcus, P. A. Branton, A. Britton, S. A. Buia, C. C. Compton, D. S. DeLuca, J. Peter-Demchok, E. T. Gelfand, P. Guan, G. E. Korzeniewski, N. C. Lockhart, C. A. Rabiner, A. K. Rao, K. L. Robinson, N. V. Roche, S. J. Sawyer, A. V. Segrè, ..., y J. Zhu. A Novel Approach to High-Quality Postmortem Tissue Procurement: The GTEx Project. *Biopreserv Biobank*, 13(5):311–319, oct. 2015. doi: 10.1089/bio.2015.0032.

J. M. Cejalvo, E. Martínez de Dueñas, P. Galván, S. García-Recio, O. Burgués Gasión, L. Paré, S. Antolín, R. Martinello, I. Blancas, B. Adamo, Á. Guerrero-Zotano, M. Muñoz, P. Nucíforo, M. Vidal, R. M. Pérez, J. I. Chacón López-Muniz, R. Caballero, V. Peg, E. Carrasco, ..., y A. Prat. Intrinsic subtypes and gene expression profiles in primary and metastatic breast cancer. *Cancer Res.*, 77(9):2213–2221, may. 2017. doi: 10.1158/0008-5472.CAN-16-2717.

J. T. Chang, E. J. Wherry, y A. W. Goldrath. Molecular regulation of effector and memory T cell differentiation. *Nat. Immunol.*, 15(12):1104–1115, dic. 2014. doi: 10.1038/ni.3031.

W. Chang, J. Cheng, J. J. Allaire, Y. Xie, y J. McPherson. *shiny: Web Application Framework for R*, 2020. R package version 1.4.0.2.

G. Chen, B. Ning, y T. Shi. Single-Cell RNA-Seq technologies and related computational data analysis. *Front Genet*, 10:317, 2019. doi: 10.3389/fgene.2019.00317.

W. Chung, H. H. Eum, H. O. Lee, K. M. Lee, H. B. Lee, K. T. Kim, H. S. Ryu, S. Kim, J. E. Lee, Y. H. Park, Z. Kan, W. Han, y W. Y. Park. Single-cell RNA-seq enables comprehensive tumour and immune cell profiling in primary breast cancer. *Nat Commun*, 8:15081, may. 2017. doi: 10.1038/ncomms15081.

G. Ciriello, M. L. Gatza, A. H. Beck, M. D. Wilkerson, S. K. Rhie, A. Pastore, H. Zhang, M. McLellan, C. Yau, C. Kandoth, R. Bowlby, H. Shen, S. Hayat, R. Fieldhouse, S. C. Lester, G. M. Tse, R. E. Factor, L. C. Collins, K. H. Allison, ..., y E. Zmuda. Comprehensive molecular portraits of invasive lobular breast cancer. *Cell*, 163(2):506–519, oct. 2015. doi: 10.1016/j.cell.2015.09.033.

I. J. Cohen y R. Blasberg. Impact of the tumor microenvironment on tumor-infiltrating lymphocytes: focus on breast cancer. *Breast Cancer (Auckl)*, 11, en. 2017. doi: 10.1177/1178223417731565.

E. J. Colbeck, A. Ager, A. Gallimore, y G. W. Jones. Tertiary lymphoid structures in cancer: drivers of antitumor immunity, immunosuppression, or bystander sentinels in disease? *Front Immunol*, 8:1830, dic. 2017. doi: 10.3389/fimmu.2017.01830.

F. S. Collins y H. Varmus. A new initiative on precision medicine. *N. Engl. J. Med.*, 372(9): 793–795, feb. 2015. doi: 10.1056/NEJMp1500523.

A. Dobin y T. R. Gingeras. Mapping RNA-seq Reads with STAR. *Curr Protoc Bioinformatics*, 51:1–11, sept. 2015. doi: 10.1002/0471250953.bi1114s51.

I. Dunham, A. Kundaje, S. F. Aldred, P. J. Collins, C. A. Davis, F. Doyle, C. B. Epstein, S. Frietze, J. Harrow, R. Kaul, J. Khatun, B. R. Lajoie, S. G. Landt, B. K. Lee, F. Pauli, K. R. Rosenbloom, P. Sabo, A. Safi, A. Sanyal, ..., y E. Birney. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57–74, sept. 2012. doi: 10.1038/nature11247.

D. Eddelbuettel y J. J. Balamuta. Extending extitR with extitC++: A Brief Introduction to extitRcpp. *PeerJ Preprints*, 5:e3188v1, ag. 2017. ISSN 2167-9843. doi: 10.7287/peerj.preprints.3188v1.

W. H. Fridman, F. Pagès, C. Sautès-Fridman, y J. Galon. The immune contexture in human tumours: impact on clinical outcome. *Nat. Rev. Cancer*, 12(4):298–306, mar. 2012. doi: 10.1038/nrc3245.

J. Galon, A. Costes, F. Sanchez-Cabo, A. Kirilovsky, B. Mlecnik, C. Lagorce-Pagès, M. Tosolini, M. Camus, A. Berger, P. Wind, F. Zinzindohoué, P. Bruneval, P. H. Cugnenc, Z. Trajanoski, W. H. Fridman, y F. Pagès. Type, density, and location of immune cells within human colorectal tumors predict clinical outcome. *Science*, 313(5795):1960–1964, sept. 2006. doi: 10.1126/science.1129139.

M. J. Gerdes, A. Sood, C. Sevinsky, A. D. Pris, M. I. Zavodszky, y F. Ginty. Emerging understanding of multiscale tumor heterogeneity. *Front Oncol*, 4:366, dic. 2014. doi: 10.3389/fonc.2014.00366.

GitHub. *GitHub: web-based hosting service for version control using Git.* GitHub, Inc., San Francisco, CA, 2018. www.github.com.

M. L. Golinski, M. Demeules, C. Derambure, G. Riou, M. Maho-Vaillant, O. Boyer, P. Joly, y S. Calbo. CD11c+ B cells are mainly memory cells, precursors of antibody secreting cells in healthy donors. *Front Immunol*, 11:32, feb. 2020. doi: 10.3389/fimmu.2020.00032.

C. Hafemeister y R. Satija. Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. *Genome Biol.*, 20(1):296, dic. 2019. doi: 10.1186/s13059-019-1874-1.

D. Hanahan y R. A. Weinberg. The hallmarks of cancer. *Cell*, 100(1):57–70, en. 2000. doi: 10.1016/s0092-8674(00)81683-9.

D. Hanahan y R. A. Weinberg. Hallmarks of cancer: the next generation. *Cell*, 144(5):646–674, mar. 2011. doi: 10.1016/j.cell.2011.02.013.

K. D. Hansen. R - S4 classes and methods, 2015. Accedido: 30-07-2020. kasperdanielhansen.github.io/genbioconductor.

G. K. Hansson y P. Libby. The immune response in atherosclerosis: a double-edged sword. *Nat. Rev. Immunol.*, 6(7):508–519, jul. 2006. doi: 10.1038/nri1882.

N. Harbeck, F. Penault-Llorca, J. Cortes, M. Gnant, N. Houssami, P. Poortmans, K. Ruddy, J. Tsang, y F. Cardoso. Breast cancer. *Nat Rev Dis Primers*, 5(1):66, sept. 2019. doi: 10.1038/s41572-019-0111-2.

T. S. Heng, M. W. Painter, K. Elpek, V. Lukacs-Kornek, N. Mauermann, S. J. Turley, D. Koller, F. S. Kim, A. J. Wagers, N. Asinovski, S. Davis, M. Fassett, M. Feuerer, D. H. Gray, S. Haxhinasto, J. A. Hill, G. Hyatt, C. Laplace, K. Leatherbee, D. Mathis, C. Benoist, R. Jianu, D. H. Laidlaw, J. A. Best, J. Knell, A. W. Goldrath, J. Jarjoura, ..., y J. Kang. The Immunological Genome Project: networks of gene expression in immune cells. *Nat. Immunol.*, 9(10):1091–1094, oct. 2008. doi: 10.1038/ni1008-1091.

J. D. Hon, B. Singh, A. Sahin, G. Du, J. Wang, V. Y. Wang, F. M. Deng, D. Y. Zhang, M. E. Monaco, y P. Lee. Breast cancer molecular subtypes: from TNBC to QNBC. *Am J Cancer Res*, 6(9):1864–1872, sept. 2016. PubMed Central: PMC5043099.

B. Jahrsdörfer, A. Vollmer, S. E. Blackwell, J. Maier, K. Sontheimer, T. Beyer, B. Mandel, O. Lunov, K. Tron, G. U. Nienhaus, T. Simmet, K. M. Debatin, G. J. Weiner, y D. Fabricius. Granzyme B produced by human plasmacytoid dendritic cells suppresses T-cell expansion. *Blood*, 115(6):1156–1165, feb. 2010. doi: 10.1182/blood-2009-07-235382.

H. Jeong, I. Hwang, S. H. Kang, H. C. Shin, y S. Y. Kwon. Tumor-associated macrophages as potential prognostic biomarkers of invasive breast cancer. *J Breast Cancer*, 22(1):38–51, mar. 2019. doi: 10.4048/jbc.2019.22.e5.

W. Kasinrerk, T. Baumruker, O. Majdic, W. Knapp, y H. Stockinger. CD1 molecule expression on human monocytes induced by granulocyte-macrophage colony-stimulating factor. *J. Immunol.*, 150(2):579–584, en. 1993. PubMed: 7678276.

J. C. Kimmel y D. R. Kelley. scnym: Semi-supervised adversarial neural networks for single cell classification. *bioRxiv*, 2020. doi: 10.1101/2020.06.04.132324. doi: 10.1101/2020.06.04.132324.

U. Klein, Y. Tu, G. A. Stolovitzky, J. L. Keller, J. Haddad, V. Miljkovic, G. Cattoretti, A. Califano, y R. Dalla-Favera. Transcriptional analysis of the B cell germinal center reaction. *Proc. Natl. Acad. Sci. U.S.A.*, 100(5):2639–2644, mar. 2003. doi: 10.1073/pnas.0437996100.

D. C. Koboldt, R. S. Fulton, M. D. McLellan, H. Schmidt, J. Kalicki-Veizer, J. F. McMichael, L. L. Fulton, D. J. Dooling, L. Ding, E. R. Mardis, R. K. Wilson, A. Ally, M. Balasundaram, Y. S. Butterfield, R. Carlsen, C. Carter, A. Chu, E. Chuah, H. J. Chun, ..., y J. D. Palchik. Comprehensive molecular portraits of human breast tumours. *Nature*, 490(7418):61–70, oct. 2012. doi: 10.1038/nature11412.

G. Kroemer, L. Senovilla, L. Galluzzi, F. André, y L. Zitvogel. Natural and therapy-induced immunosurveillance in breast cancer. *Nat. Med.*, 21(10):1128–1138, oct. 2015. doi: 10.1038/nm.3944.

M. V. Kuleshov, M. R. Jones, A. D. Rouillard, N. F. Fernandez, Q. Duan, Z. Wang, S. Koplev, S. L. Jenkins, K. M. Jagodnik, A. Lachmann, M. G. McDermott, C. D. Monteiro, G. W. Gundersen, y A. Ma'ayan. Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. *Nucleic Acids Res.*, 44(1):90–97, jul. 2016. doi: 10.1093/nar/gkw377.

J. H. Levine, E. F. Simonds, S. C. Bendall, K. L. Davis, e. l. A. D. Amir, M. D. Tadmor, O. Litvin, H. G. Fienberg, A. Jager, E. R. Zunder, R. Finck, A. L. Gedman, I. Radtke, J. R. Downing, D. Pe'er, y G. P. Nolan. Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. *Cell*, 162(1):184–197, jul. 2015. doi: 10.1016/j.cell.2015.05.047.

B. Li y C. N. Dewey. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC Bioinformatics*, 12:323, ag. 2011. doi: 10.1186/1471-2105-12-323.

B. Li, J. S. Liu, y X. S. Liu. Revisit linear regression-based deconvolution methods for tumor gene expression data. *Genome Biol.*, 18(1):127, jul. 2017. doi: 10.1186/s13059-017-1256-5.

L. I. Lin. A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, 45(1): 255–268, mar. 1989. PubMed: 2720055.

A. Lun y D. Risso. *SingleCellExperiment: S4 Classes for single cell data*, 2019. Versión 1.16.1. www.bioconductor.org/packages/release/bioc/html/SummarizedExperiment.html.

A. T. L. Lun, H. Pagès, y M. L. Smith. beachmat: A Bioconductor C++ API for accessing high-throughput biological data from a variety of R matrix types. *PLoS Comput. Biol.*, 14 (5):e1006135, may. 2018. doi: 10.1371/journal.pcbi.1006135.

N. A. Mabbott, J. K. Baillie, H. Brown, T. C. Freeman, y D. A. Hume. An expression atlas of human primary cells: inference of gene function from coexpression networks. *BMC Genomics*, 14:632, sept. 2013. doi: 10.1186/1471-2164-14-632.

J. H. Martens y H. G. Stunnenberg. BLUEPRINT: mapping human blood cell epigenomes. *Haematologica*, 98(10):1487–1489, oct. 2013. doi: 10.3324/haematol.2013.094243.

M. Martin. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal*, 17(1):10–12, may. 2011. doi: 10.14806/ej.17.1.200.

D. J. McCarthy, K. R. Campbell, A. T. Lun, y Q. F. Wills. Scater: pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R. *Bioinformatics*, 33 (8):1179–1186, apr. 2017. doi: 10.1093/bioinformatics/btw777.

N. McGranahan y C. Swanton. Biological and therapeutic impact of intratumor heterogeneity in cancer evolution. *Cancer Cell*, 27(1):15–26, en. 2015. doi: 10.1016/j.ccell.2014.12.001.

S. Mohammadi, N. Zuckerman, A. Goldsmith, y A. Grama. A critical survey of deconvolution methods for separating cell types in complex tissues. *Proceedings of the IEEE*, 105(2):340366, feb. 2017. ISSN 1558-2256. doi: 10.1109/jproc.2016.2607121.

G. Monaco, B. Lee, W. Xu, S. Mustafah, Y. Y. Hwang, C. Carré, N. Burdin, L. Visan, M. Ceccarelli, M. Poidinger, A. Zippelius, J. Pedro de Magalhães, y A. Larbi. RNA-Seq signatures normalized by mRNA abundance allow absolute deconvolution of human immune cell types. *Cell Rep*, 26(6):1627–1640, feb. 2019. doi: 10.1016/j.celrep.2019.01.041.

T. A. Moo, R. Sanford, C. Dang, y M. Morrow. Overview of breast cancer therapy. *PET Clin*, 13(3):339–354, jul. 2018. doi: 10.1016/j.cpet.2018.02.006.

M. Morgan, V. Obenchain, J. Hester, y H. Pagès. *SummarizedExperiment: SummarizedExperiment container*, 2019. R package version 1.16.1.

N. E. Navin. The first five years of single-cell cancer genomics and beyond. *Genome Res.*, 25 (10):1499–1507, oct. 2015. doi: 10.1101/gr.191098.115.

A. M. Newman, C. B. Steen, C. L. Liu, A. J. Gentles, A. A. Chaudhuri, F. Scherer, M. S. Khodadoust, M. S. Esfahani, B. A. Luca, D. Steiner, M. Diehn, y A. A. Alizadeh. Determining cell type abundance and expression from bulk tissues with digital cytometry. *Nat. Biotechnol.*, 37(7):773–782, jul. 2019. doi: 10.1038/s41587-019-0114-2.

H. Pagès. *HDF5Array: HDF5 backend for DelayedArray objects*, 2020. R package version 1.14.4.

H. Pagès, P. with contributions from Hickey, y A. Lun. *DelayedArray: A unified framework for working transparently with on-disk and in-memory array-like datasets*, 2020. Versión 0.12.3. www.bioconductor.org/packages/release/bioc/html/DelayedArray.html.

C. M. Perou, T. Sorlie, M. B. Eisen, M. van de Rijn, S. S. Jeffrey, C. A. Rees, J. R. Pollack, D. T. Ross, H. Johnsen, L. A. Akslen, Ø. Fluge, A. Pergamenschikov, C. Williams, S. X. Zhu, P. E. Lønning, A. L. Børresen-Dale, P. O. Brown, y D. Botstein. Molecular portraits of human breast tumours. *Nature*, 406(6797):747–752, ag. 2000. doi: 10.1038/35021093.

S. Q. Qiu, S. J. H. Waaijer, M. C. Zwager, E. G. E. de Vries, B. van der Vegt, y C. P. Schröder. Tumor-associated macrophages in breast cancer: innocent bystander or important player? *Cancer Treat. Rev.*, 70:178–189, nov. 2018. doi: 10.1016/j.ctrv.2018.08.010.

X. Qiu, A. Hill, J. Packer, D. Lin, Y. A. Ma, y C. Trapnell. Single-cell mRNA quantification and differential analysis with census. *Nat. Methods*, 14(3):309–315, mar. 2017. doi: 10.1038/nmeth.4150.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. Versión 3.6.0. www.R-project.org.

D. Risso, F. Perraudeau, S. Gribkova, S. Dudoit, y J. P. Vert. A general and flexible method for signal extraction from single-cell RNA-seq data. *Nat Commun*, 9(1):284, en. 2018. doi: 10.1038/s41467-017-02554-5.

RStudio Team. *RStudio: Integrated Development Environment for R.* RStudio, Inc., Boston, MA, 2016. Versión 1.1.453. www.rstudio.com.

A. J. Ruiz-Alcaraz, V. Carmona-Martínez, M. Tristán-Manzano, F. Machado-Linde, M. L. Sánchez-Ferrer, P. García-Peñarrubia, y M. Martínez-Esparza. Characterization of human peritoneal monocyte/macrophage subsets in homeostasis: Phenotype, GATA6, phagocytic/oxidative activities and cytokines expression. *Sci Rep*, 8(1):12794, ag. 2018. doi: 10.1038/s41598-018-30787-x.

R. Salgado, C. Denkert, S. Demaria, N. Sirtaine, F. Klauschen, G. Pruneri, S. Wienert, G. Van den Eynden, F. L. Baehner, F. Penault-Llorca, E. A. Perez, E. A. Thompson, W. F. Symmans, A. L. Richardson, J. Brock, C. Criscitiello, H. Bailey, M. Ignatiadis, G. Floris, ..., y S. Loi. The evaluation of tumor-infiltrating lymphocytes (TILs) in breast cancer: recommendations by an international TILs working group 2014. *Ann. Oncol.*, 26(2):259–271, feb. 2015. doi: 10.1093/annonc/mdu450.

A. Sarvaria, J. A. Madrigal, y A. Saudemont. B cell regulation in cancer and anti-tumor immunity. *Cell. Mol. Immunol.*, 14(8):662–674, ag. 2017. doi: 10.1038/cmi.2017.35.

P. Savas, B. Virassamy, C. Ye, A. Salim, C. P. Mintoff, F. Caramia, R. Salgado, D. J. Byrne, Z. L. Teo, S. Dushyanthen, A. Byrne, L. Wein, S. J. Luen, C. Poliness, S. S. Nightingale, A. S. Skandarajah, D. E. Gyorki, C. M. Thornton, P. A. Beavis, ..., y S. Loi. Single-cell profiling of breast cancer T cells reveals a tissue-resident memory subset associated with improved prognosis. *Nat. Med.*, 24(7):986–993, jul. 2018. doi: 10.1038/s41591-018-0078-7.

B. Schloerke, D. Cook, J. Larmarange, F. Briatte, M. Marbach, E. Thoen, A. Elberg, y J. Crowley. *GGally: Extension to 'ggplot2'*, 2020. Versión 2.0.0. CRAN.R-project.org/package=GGally.

J. A. Seidel, A. Otsuka, y K. Kabashima. Anti-PD-1 and Anti-CTLA-4 therapies in cancer: mechanisms of action, efficacy, and limitations. *Front Oncol*, 8:86, mar. 2018. doi: 10.3389/fonc.2018.00086.

T. J. Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *Proc. Natl. Acad. Sci. U.S.A.*, en. 2020. doi: 10.1073/pnas.1907373117.

J. S. Seo, J. W. Lee, A. Kim, J. Y. Shin, Y. J. Jung, S. B. Lee, Y. H. Kim, S. Park, H. J. Lee, I. K. Park, C. H. Kang, J. Y. Yun, J. Kim, y Y. T. Kim. Whole exome and transcriptome analyses integrated with microenvironmental immune signatures of lung squamous cell carcinoma. *Cancer Immunol Res*, 6(7):848–859, jul. 2018. doi: 10.1158/2326-6066.CIR-17-0453.

O. Stegle, S. A. Teichmann, y J. C. Marioni. Computational and analytical challenges in single-cell transcriptomics. *Nat. Rev. Genet.*, 16(3):133–145, mar. 2015. doi: 10.1038/nrg3833.

T. Stuart, A. Butler, P. Hoffman, C. Hafemeister, E. Papalexi, W. M. Mauck III, Y. Hao, M. Stoeckius, P. Smibert, y R. Satija. Comprehensive integration of single-cell data. *Cell*, 177:1888–1902, jun. 2019. doi: 10.1016/j.cell.2019.05.031.

T. Sørlie, R. Tibshirani, J. Parker, T. Hastie, J. S. Marron, A. Nobel, S. Deng, H. Johnsen, R. Pesich, S. Geisler, J. Demeter, C. M. Perou, P. E. Lønning, P. O. Brown, A. L. Børresen-Dale, y D. Botstein. Repeated observation of breast tumor subtypes in independent gene expression data sets. *Proc. Natl. Acad. Sci. U.S.A.*, 100(14):8418–8423, jul. 2003. doi: 10.1073/pnas.0932692100.

C. Torroja y F. Sanchez-Cabo. Digitaldlsorter: Deep-Learning on scRNA-Seq to deconvolute gene expression data. *Front Genet*, 10:978, oct. 2019. doi: 10.3389/fgene.2019.00978.

K. Ushey, J. J. Allaire, y Y. Tang. *reticulate: Interface to 'Python'*, 2020. R package version 1.16.

T. P. van den Bosch, K. Caliskan, M. D. Kraaij, A. A. Constantinescu, O. C. Manintveld, P. J. Leenen, J. H. von der Thüsen, M. C. Clahsen-van Groningen, C. C. Baan, y A. T. Rowshani. CD16+ monocytes and skewed macrophage polarization toward M2 type hallmark heart transplant acute Cellular Rejection. *Front Immunol*, 8:346, mar. 2017. doi: 10.3389/fimmu.2017.00346.

G. Van Rossum y F. L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

G. P. Wagner, K. Kin, y V. J. Lynch. Measurement of mRNA abundance using RNA-seq data: RPKM measure is inconsistent among samples. *Theory Biosci.*, 131(4):281–285, dic. 2012. doi: 10.1007/s12064-012-0162-3.

H. Wickham. testthat: Get started with testing. *The R Journal*, 3:5–10, 2011. Versión 2.3.1. www.journal.r-project.org.

H. Wickham. *R Packages*. O'Reilly Media, Inc., 1st edition, 2015. ISBN 1491910593.

H. Wickham. *Advanced R*. CRC Press, Boca Raton, Florida, 2016a. ISBN 1-4987-5980-7.

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016b. ISBN 978-3-319-24277-4. Versión 3.2.1. www.ggplot2.tidyverse.org.

H. Wickham y J. Bryan. *usethis: Automate Package and Project Setup*, 2020. Versión 1.6.1. www.CRAN.R-project.org/package=usethis.

H. Wickham, P. Danenberg, G. Csárdi, y M. Eugster. *roxygen2: In-Line Documentation for R*, 2020a. Versión 7.1.1. www.CRAN.R-project.org/package=roxygen2.

H. Wickham, J. Hester, y W. Chang. *devtools: Tools to Make Developing R Packages Easier*, 2020b. Versión 2.3.1. www.CRAN.R-project.org/package=devtools.

R. Wilentzik Müller y I. Gat-Viks. Exploring Neural Networks and Related Visualization Techniques in Gene Expression Data. *Front Genet*, 11:402, may. 2020. doi: 10.3389/fgene.2020.00402.

J. Yang, L. Zhang, C. Yu, X. F. Yang, y H. Wang. Monocyte and macrophage differentiation: circulation inflammatory monocyte as biomarker for inflammatory diseases. *Biomark Res*, 2 (1):1, en. 2014. doi: 10.1186/2050-7771-2-1.

X. Yang. Multitissue multiomics systems biology to dissect complex diseases. *Trends Mol Med*, 26(8):718–728, ag. 2020. doi: 10.1016/j.molmed.2020.04.006.

X. Yu, L. Zhang, A. Chaudhry, A. S. Rapaport, y W. Ouyang. Unravelling the heterogeneity and dynamic relationships of tumor-infiltrating T cells by single-cell RNA sequencing analysis. *J. Leukoc. Biol.*, 107(6):917–932, jun. 2020. doi: 10.1002/JLB.6MR0320-234R.

L. Zappia, B. Phipson, y A. Oshlack. Splatter: simulation of single-cell RNA sequencing data. *Genome Biol.*, 18(1):174, sept. 2017. doi: 10.1186/s13059-017-1305-0.

A

Material suplementario: Información sobre el código implementado

El repositorio de GitHub sobre el que se ha desarrollado el paquete y en el que puede encontrarse tanto el código como la documentación y la viñeta desarrolladas es diegommcc/digitalDLSorteR. En el README, se muestran las indicaciones para su instalación, así como documentación con información sobre la instalación de Keras en el entorno de R. En cualquier caso, a continuación se muestra el código necesario para la instalación:

```
if (!requireNamespace("devtools", quietly = TRUE))
    install.packages("devtools")

devtools::install_github("diegommcc/digitalDLSorteR")
```

Code Box A.1: Pasos para la instalación de digitalDLSorteR.

A pesar de que el paquete digitalDLSorteR continúa siendo una versión aún en desarrollo, todas las funcionalidades expuestas a lo largo del presente Trabajo Fin de Máster están disponibles.

Además, es posible visitar la documentación en formato pdf y la viñeta en formato html en el repositorio diegommcc/digitalDLSorteR_Documentation. Estos ficheros son los equivalentes a los Rd del directorio digitalDLSorteR/man del paquete y a su versión como comentarios en el propio código mediante el uso del paquete roxygen2.

B

Material suplementario: Figuras adicionales

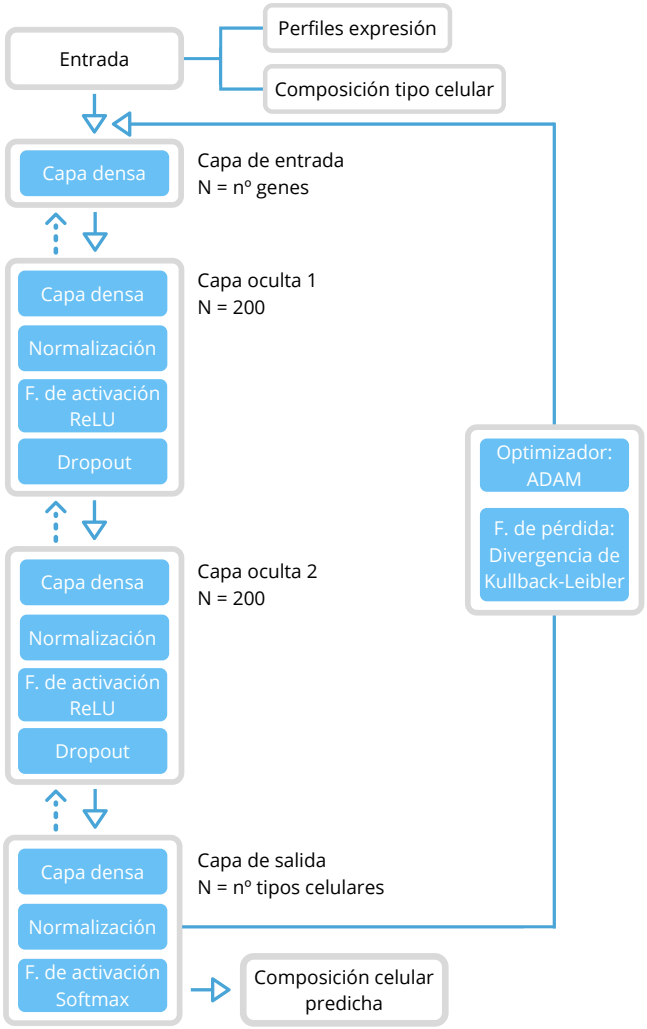

Figura B.1: Diagrama de la arquitectura de la Red Neuronal Profunda implementada en digitalDLSorteR.

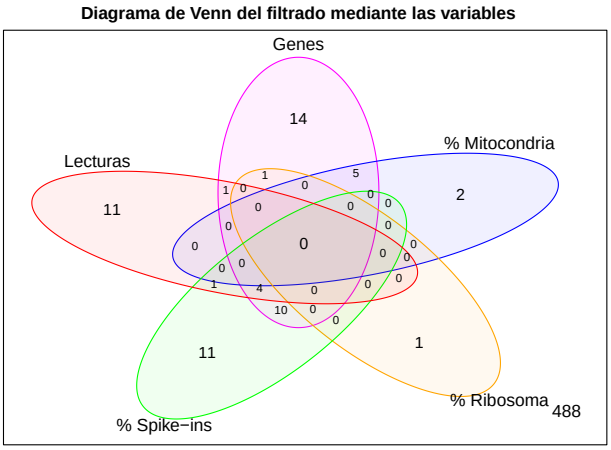

Figura B.2: Diagrama de Venn del número de células filtradas mediante las diferentes variables utilizadas.

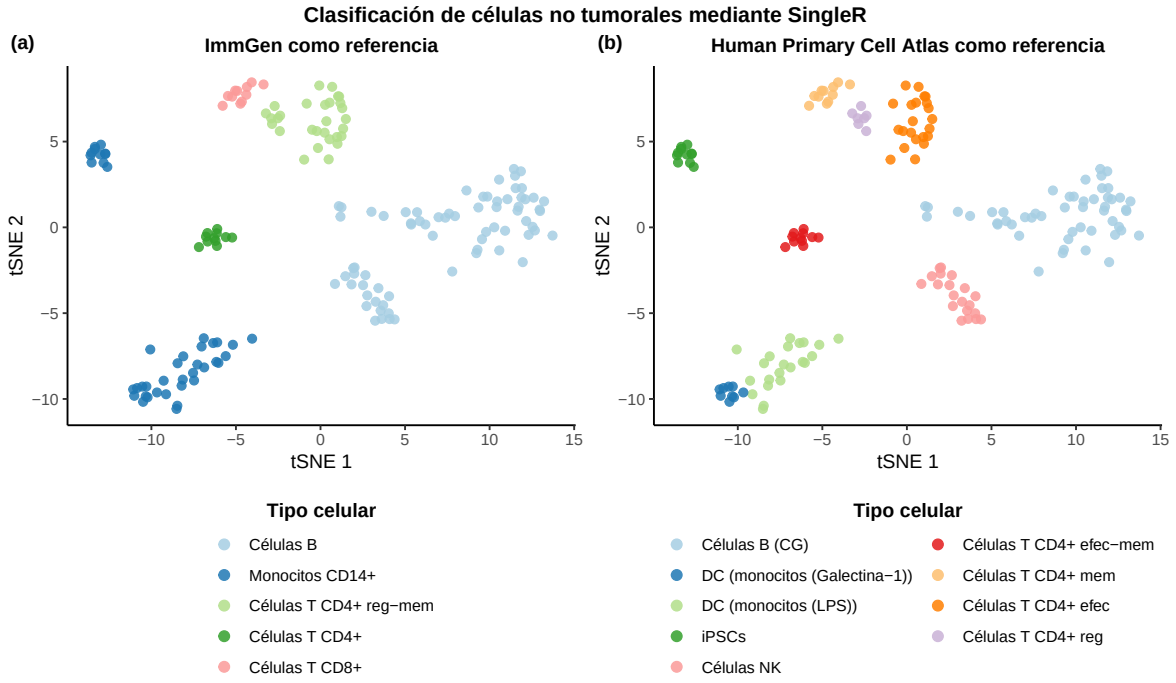

Figura B.3: Clasificación determinada con SingleR de los clústeres de las células no tumorales. **(a):** Utilizando ImmGen como referencia. **(b):** Utilizando Human Primary Cell Atlas como referencia.

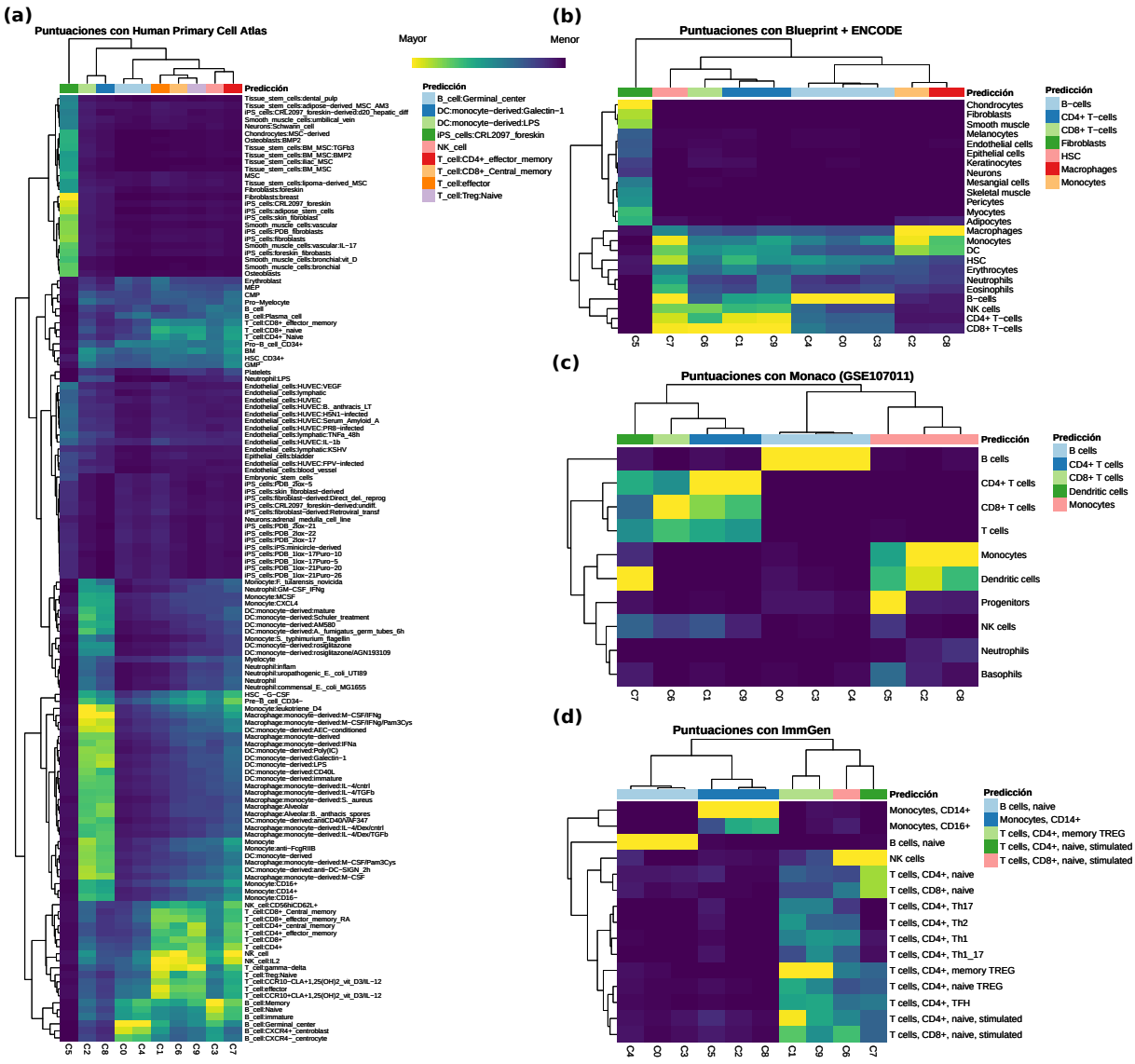

Figura B.4: Mapas de calor correspondientes a las puntuaciones obtenidas por cada clúster durante su identificación con SingleR antes del refinamiento. **(a):** Puntuaciones obtenidas con Human Primary Cell Atlas como referencia. **(b):** Puntuaciones obtenidas con Blueprint + ENCODE como referencia. **(c):** Puntuaciones obtenidas con GSE107011 como referencia. **(d):** Puntuaciones obtenidas con ImmGen como referencia.

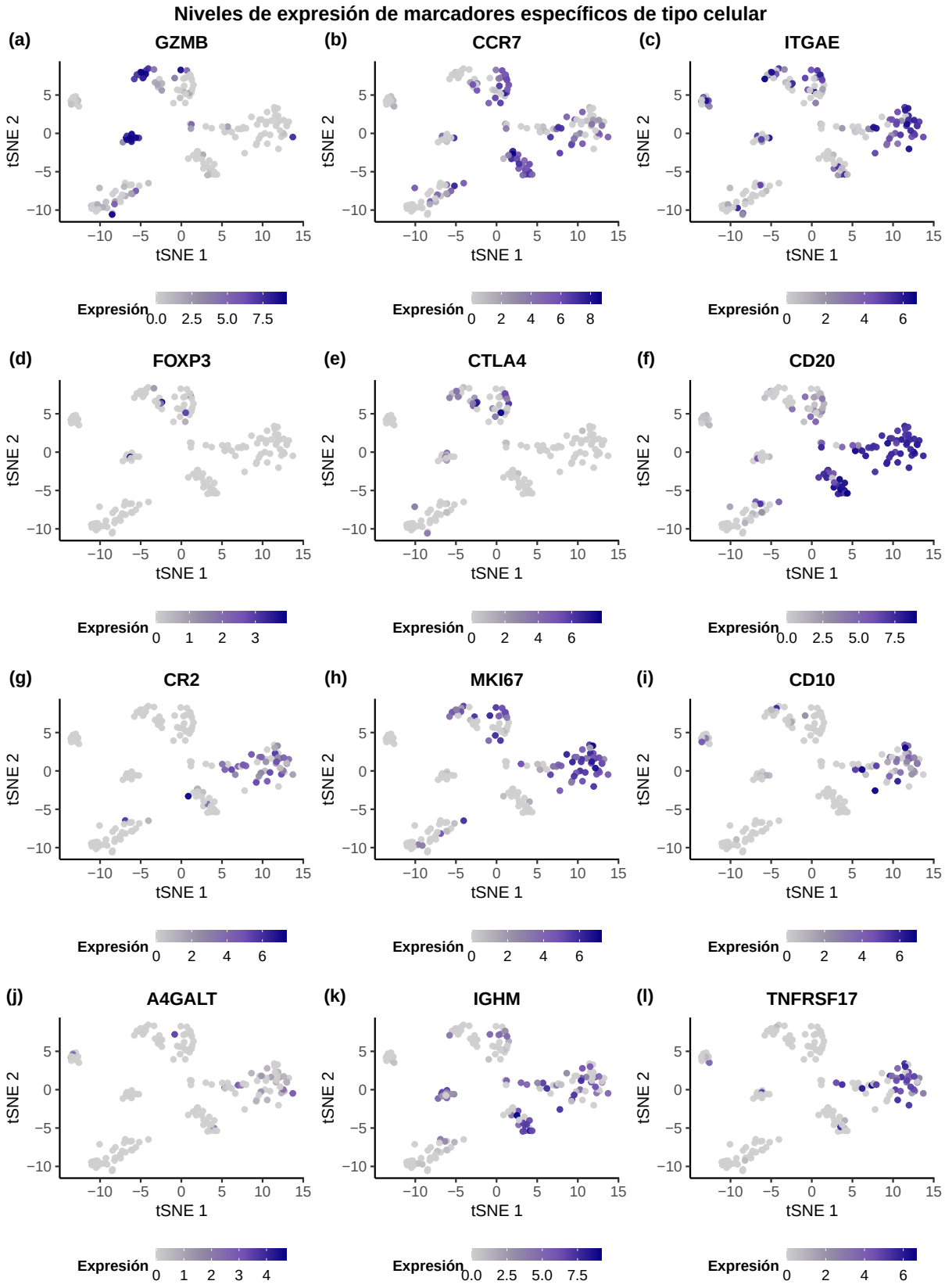

Figura B.5: Representación tSNE de los niveles de expresión de los marcadores utilizados para la caracterización manual de las células no tumorales. El título de cada gráfico se corresponde con el gen que se está representando.

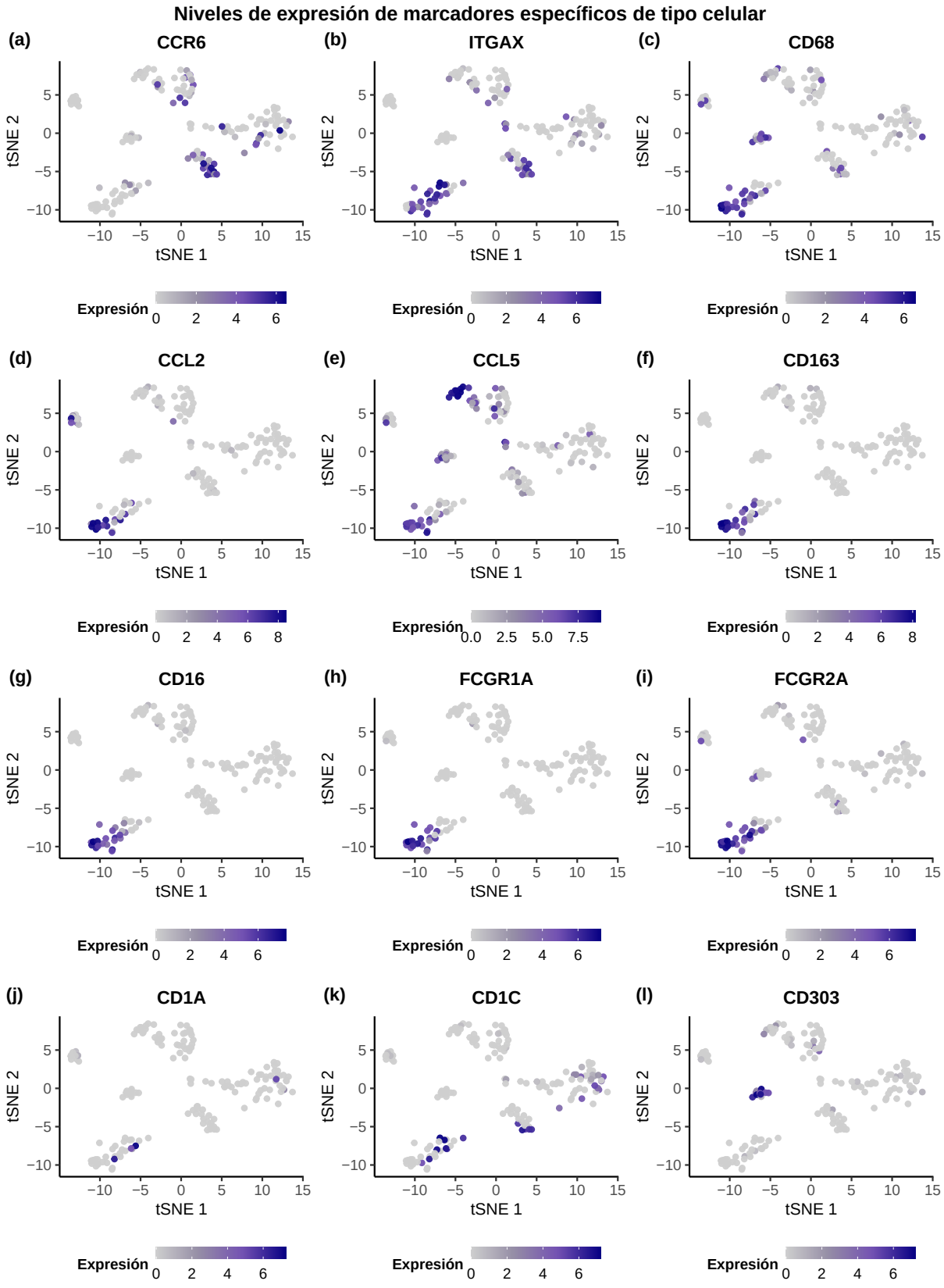

Figura B.6: Representación tSNE de los niveles de expresión de los marcadores utilizados para la caracterización manual de las células no tumorales. El título de cada gráfico se corresponde con el gen que se está representando.

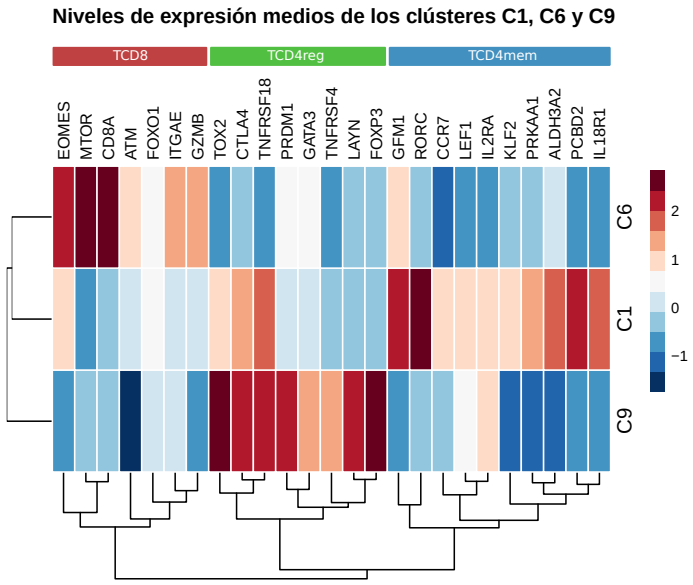

Figura B.7: Mapa de calor de los niveles de expresión medios de los clústeres C1, C6 y C9. Genes marcadores de células T CD8+, células T CD4+ reguladoras y células T CD4+ de memoria.

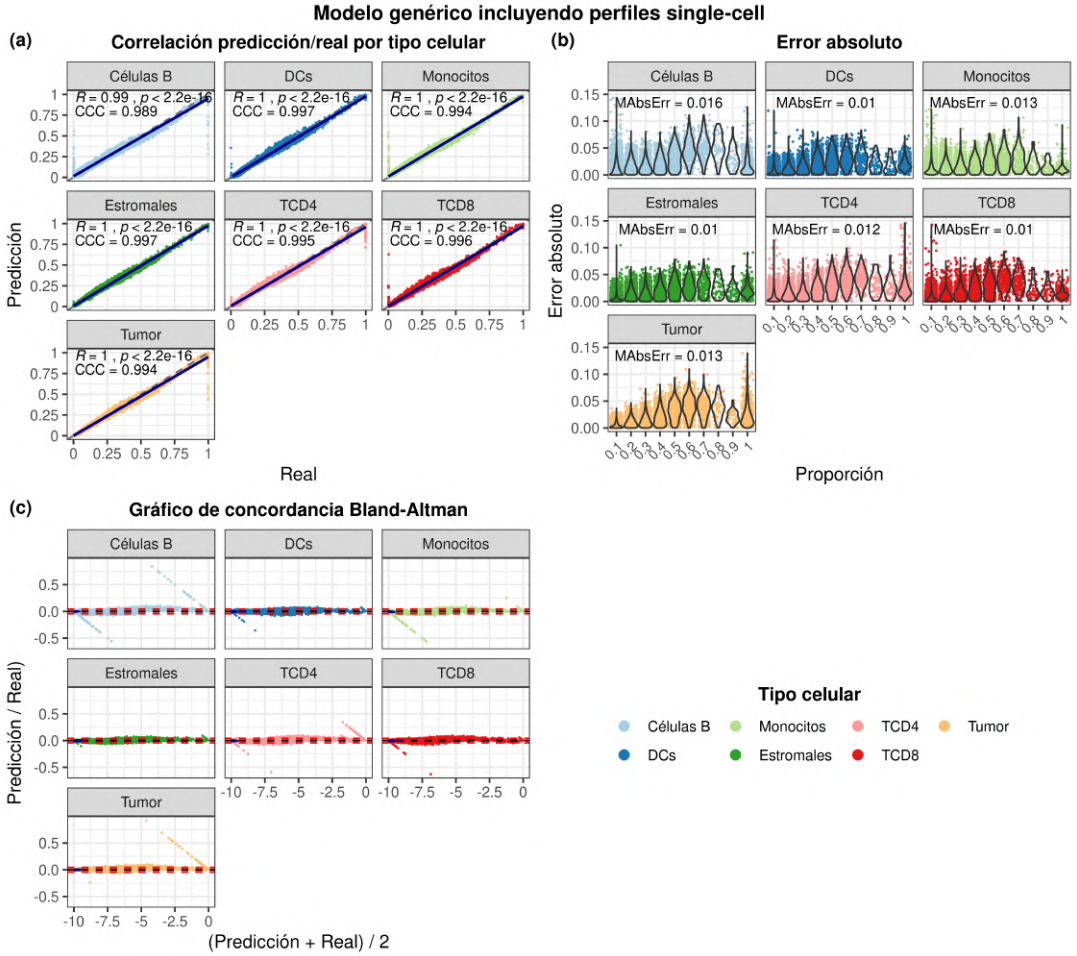

Figura B.9: Resultados del modelo genérico sobre el conjunto de datos de test incluyendo perfiles *single-cell*. **(a):** Gráficos de correlación predicción/proporciones reales por tipo celular (línea sólida: recta ajustada; línea discontinua: identidad). **(b):** Distribución del error absoluto por tipo celular y por agrupaciones de 0,1 de las probabilidades (MAbsErr: error medio absoluto por tipo celular). **(c):** Gráficos de concordancia Bland-Altman (líneas rojas discontinuas: $\pm$ 1,96 $*$ desviación estándar sobre la media; línea discontinua negra: media; curvas discontinuas azules: densidad de puntos).

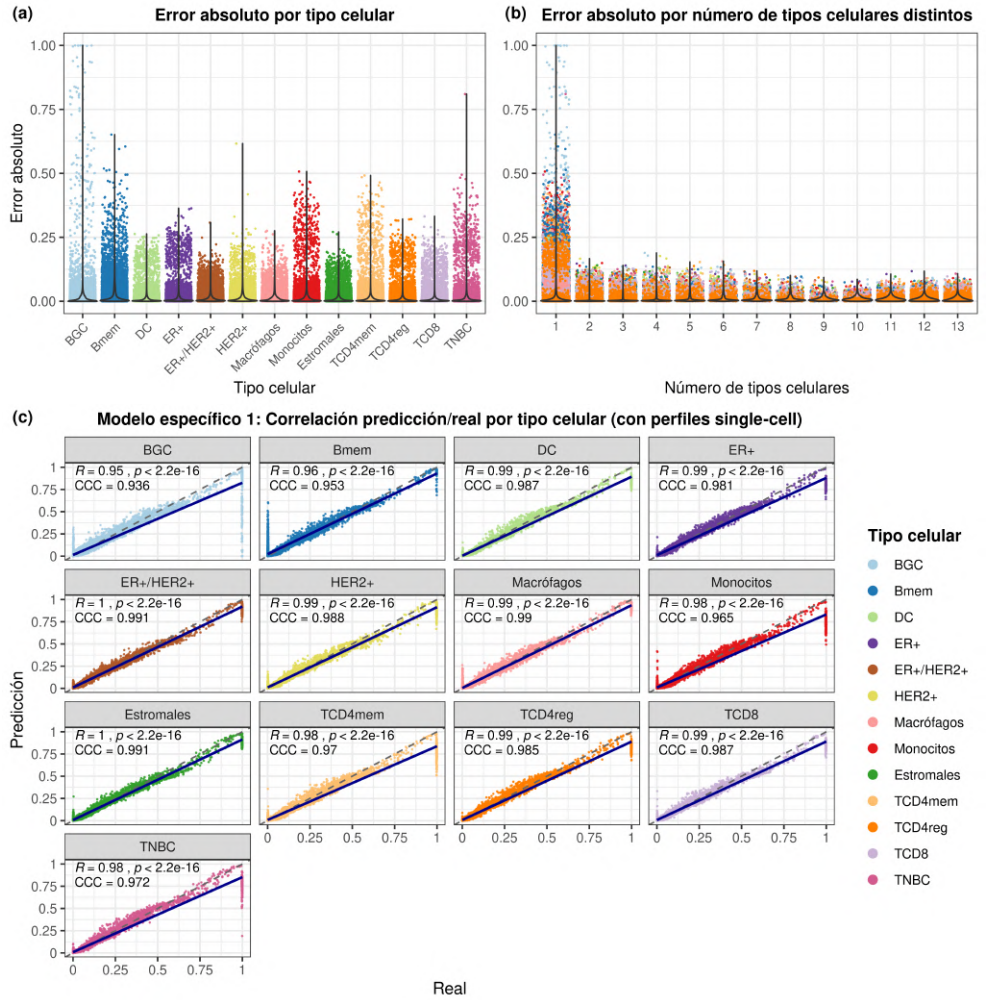

Figura B.8: Resultados del modelo específico 1 sobre el conjunto de datos de test incluyendo perfiles *single-cell*. **(a):** *Violin plots* del error absoluto por tipo celular. **(b):** *Violin plots* del error absoluto por número de tipos celulares distintos. **(c):** Gráficos de correlación predicción/proporción real por tipo celular (línea sólida: recta ajustada; línea discontinua: identidad).

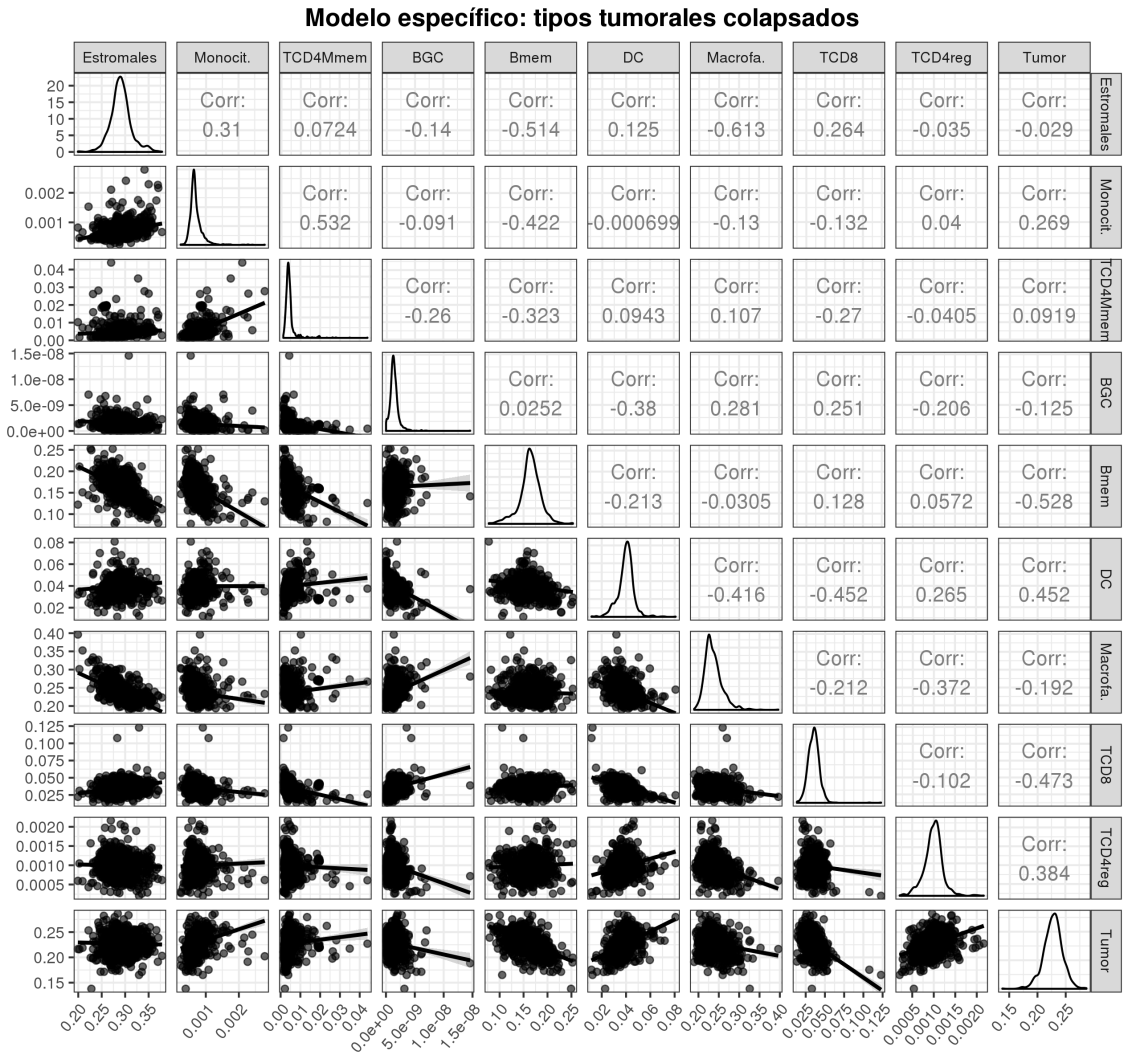

Figura B.10: Matriz de correlación de las predicciones de los tipos celulares considerados por el modelo tras el colapso de los tipos tumorales ER+, HER2+, ER+/HER2+ y TNCB en el grupo 'tumor' mediante el argumento `simplify.set`.